

New Algebra: groups, rings, modules and fields with carrier sets

Lawrence C. Paulson

Abstract

A development of abstract algebra in which every structure is a *carrier set* together with its operations, following the locale-based style of Ballarín's *Basic Algebra* [1], itself after Jacobson [2]. A group is a set with a composition and a unit; a ring is a set with two operations; an extension is a carrier with its own addition and multiplication. Nothing is a record, and no structure is identified with a type, so substructures, quotients and extensions are all ordinary values that can be quantified over, ordered, and exhausted by transfinite arguments.

The group theory covers subgroup lattices, normal series, the Zassenhaus and Schreier refinement theorems, the Jordan–Hölder uniqueness theorem, finite and indexed products, group actions, and solvability. The ring theory covers ideals, quotients, the Chinese remainder theorem for finite families, Krull's theorem on the existence of maximal ideals, principal ideal domains, Euclidean domains, unique factorisation, and polynomials in one and in indexed families of indeterminates. The module theory covers submodules, exact sequences, complements, free modules and their universal property, and finite-dimensional vector spaces with the Steinitz exchange lemma and a well-defined dimension.

On these rest the field theory and Galois theory: minimal polynomials, the degree of a finite extension and the tower law, transitivity of algebraicity, normality and separability, Artin's theorem, the Galois correspondence, normal closures, the primitive element theorem, and the finite fields. The cyclicity of the multiplicative group of a finite field is proved, along with the residue rings $\mathbb{Z}/n\mathbb{Z}$, their unit groups, and Euler's and Fermat's theorems.

The development is self-contained: it depends only on the Isabelle/HOL distribution, and is intended to supersede the record-based `HOL-Algebra` rather than to build upon it. Importing the theory `Algebra` makes the whole of it available.

Contents

0.1	Compatibility with Ballarín's original development	17
1	Monoids and Groups	23

1.1	Monoids of Transformations and Abstract Monoids	23
1.2	Groups of Transformations and Abstract Groups	24
1.3	The subgroup generated by a set	29
1.4	Isomorphisms. Cayley's Theorem	32
1.5	Generalized Associativity. Commutativity	38
1.6	Orbits. Cosets of a Subgroup	39
1.7	Congruences. Quotient Monoids and Groups	43
1.8	Homomorphisms	49
1.9	Correspondence Theorem and Third Isomorphism Theorem . . .	57
1.10	Dropping from elements of a partition to representatives	60
1.11	Abstract type of monoids	63
2	Partial orders, lattices and complete lattices	64
2.1	Partial orders	64
2.1.1	Bounds	65
2.1.2	Suprema and infima	65
2.2	Lattices	66
2.2.1	Lattice laws	67
2.3	Complete lattices	68
2.4	Isotone maps	69
2.5	Galois connections	69
3	The lattice of subgroups	70
3.1	Intersections of subgroups	71
3.2	The complete lattice of subgroups	71
4	Isomorphism classes of set-based monoids	72
4.1	The trivial isomorphism class	73
5	Finite normal series	73
6	Finite relative normal chains	76
6.1	Isomorphism classes of normal-series factors	76
6.2	Normal factors	77
6.3	Isomorphism classes of normal-series factors	79
7	Group Actions	79
7.1	Definition and Basic Properties	80
7.2	Stabilizer	81
7.3	Fixed Points	81

7.4	Orbits	82
7.5	The Orbit-Stabilizer Theorem	83
7.6	Connection to Transformation Groups	83
7.7	The permutation representation	84
7.8	Orbit Partition	85
8	Sylow's Theorem	87
8.1	Cyclic subgroups	89
9	The First Sylow Theorem	90
9.1	Conjugation, Center, and Centralizer	93
9.2	First Sylow Theorem	96
10	The Second Sylow Theorem	96
11	Commutators and the Derived Subgroup	101
11.1	The Derived Series and Solvable Groups	103
12	Transfer of Solvability	104
12.1	Subgroups	106
12.2	Solvability of an extension	107
13	The external direct product of two groups	107
13.1	Projections are homomorphisms; the derived series is componentwise-bounded	108
13.2	Product of solvable groups is solvable	109
14	The direct product of an indexed family of groups	109
14.1	The product locale	109
14.2	Carrier, composition and unit	110
14.3	The product is a group	110
14.4	Projections are homomorphisms	111
14.5	The product of a family of abelian groups is abelian	111
15	The Zassenhaus Lemma	112
15.1	The quotient isomorphism	114
16	The Schreier refinement theorem	116
16.1	Refining pairs of finite normal series	116
16.2	Refinement rows	119
16.3	Factor-class refinements	120

16.4	Reduced Schreier refinements	121
17	Composition series, etc.	123
17.1	Simple groups	123
17.2	Normal chains inside a simple factor	125
18	The Jordan–Hölder theorem	126
19	Maximal normal subgroups	128
19.1	The simple-quotient characterisation	128
19.2	Existence in finite groups	128
20	Composition series of finite groups	129
20.1	Existence	129
21	Solvability and composition factors	130
21.1	Auxiliary group-theoretic bridges	130
21.2	The factor criterion	130
21.3	Independence and finite-group formulations	131
22	The Symmetric and Alternating Groups	132
22.1	The symmetric group as a <i>Group</i>	132
22.2	Relation to the development’s abstract symmetric group	133
22.3	The sign homomorphism and the alternating group	133
22.4	Three-cycles	134
23	Insolvability of the Symmetric Group	134
23.1	Three-cycles as commutators	135
23.2	Persistence of three-cycles in the derived series	135
23.3	The main theorem	136
23.4	The alternating group	136
24	Generating the Symmetric Group	137
24.1	A general lemma on generated subgroups	137
24.2	Conjugation is an automorphism of the generated subgroup	137
24.3	Transpositions generate the symmetric group	138
24.4	The standard n -cycle	138
24.5	Conjugating a transposition	139
24.6	The symmetric group is generated by a transposition and a cycle	139
24.7	A transposition and a p -cycle generate S_p for prime p	141
24.8	An arbitrary cycle and transposition generate S_p	143

24.9	An order- p element of S_p is a p -cycle	143
25	Composition factors of the symmetric and alternating groups	145
25.1	A finite group has a composition series, and S_5 in particular . . .	146
25.2	The criterion is consistent with S_n being unsolvable	146
26	Subfields and Field Homomorphisms (Type-Class Setting)	146
26.1	The subfield generated by a set	147
26.2	The subfield locale	147
26.3	Field homomorphisms on a subfield	148
27	Polynomials over a Subfield; Minimal Polynomials in a Tower	150
27.1	Polynomials with coefficients in a subfield	150
27.2	Division by a monic polynomial over a subfield	151
27.3	Minimal polynomial over a subfield	152
28	Simple Algebraic Extensions are Subfields	154
29	Extending a Field Homomorphism across a Simple Algebraic Extension	156
29.1	The mapped polynomial <i>map-poly f</i> on <i>poly-over K</i>	156
29.2	Evaluation commutes with the homomorphism	157
29.3	Well-definedness of the extension	157
29.4	The extension theorem	157
30	Rings	158
30.1	Definition and Elementary Properties	158
30.2	Commutative rings, integral domains, and fields	159
30.3	Ideals, Quotient Rings	160
30.4	Homomorphisms of Rings. Basic Theorems	162
30.5	Abstract type of rings	166
30.6	Linking the locale-based field theory to Isabelle's type classes . .	168
31	Generic field automorphisms and fixed fields	169
31.1	Automorphisms are additive, multiplicative and fix the fixed field	169
32	Subfields of the Complex Numbers and their Automorphisms	173
32.1	Subfields of $\mathbb{C}$	173
32.2	Field automorphisms and the Galois group	174
32.3	Complex conjugation as an automorphism	176
32.4	The subfield generated by a set; splitting subfields	176

32.5 Faithfulness of the action on a generating set	177
33 Iterating the Iso-Extension over a Finite Set of Algebraic Elements	178
34 Transitivity of the Galois Action on the Roots	179
34.1 Rational Subfields and Irreducibility over the Rationals	181
35 The Galois Group Acts on the Roots	182
36 Embedding a Faithful Finite Action into a Concrete Symmetric Group	184
36.1 Relabelling Symmetric Groups along a Bijection	184
37 The Symmetric Group on the Roots and Non-Solvability	186
38 Finite p-groups	186
39 The subring generated by a set	187
39.1 Intersections of subrings	187
39.2 The generated subring	188
40 The characteristic of a ring; finite domains have prime characteristic	188
41 Product Operator for Commutative Monoids	190
41.1 Products over Finite Sets	190
41.2 Results for Abelian Groups	193
41.3 Transfer between two commutative monoids	194
42 The Polynomial Ring $R[X]$ (Carrier-Set Construction)	195
42.1 Carrier, coefficients, and support	195
42.2 Zero and one	195
42.3 Addition	196
42.4 Multiplication (convolution)	196
42.5 Closure under the operations	196
42.6 Distribution of multiplication over finite additive sums	197
42.7 Pointwise additive structure	197
42.8 Multiplicative unit	198
42.9 Distributivity	199
42.10A Fubini law for the additive finite sum	199
42.11 Associativity of convolution	199

42.12	Coefficients, degree, monomials, constants, and the variable . . .	199
42.13	Polynomials of bounded degree	200
42.14	The ring structure	204
42.15	Evaluation at a point	204
42.16	Evaluation along a ring homomorphism	206
43	Polynomials over a Commutative Ring (continued): evaluation is multiplicative	207
44	Polynomials over a Commutative Ring	208
45	Polynomials over an Integral Domain	208
46	Polynomial Division over a Field	209
46.1	Units of $F[X]$	209
46.2	Remainder modulo a polynomial	211
46.3	The factor theorem	213
46.4	A nonzero polynomial has at most <i>degree</i> roots	213
46.5	Low-degree irreducibility	214
46.6	Factorisation into irreducibles	214
46.7	Splitting into linear factors	215
46.8	Monic polynomials and normalisation	216
46.9	The polynomial ring over a field is a principal ideal domain . . .	217
47	Pushing a Polynomial along a Ring Homomorphism	217
48	Prime and Maximal Ideals	220
49	Principal Ideals	223
49.1	Existence of maximal ideals	223
50	Ideals of the Polynomial Ring; Kronecker's Construction	225
50.1	The size of the quotient ring	226
50.2	Divisibility and Euclid's lemma	227
50.3	Uniqueness of the irreducible factorisation	228
51	The prime fields $GF(p) = \mathbb{Z}/p\mathbb{Z}$	228
51.1	The residue rings $\mathbb{Z}/n\mathbb{Z}$	229
51.2	The residue ring as a field	230
51.3	The two-element field as a non-vacuity witness	231
51.4	The two-element field with polynomial machinery	232

51.5	The 3-element prime field with polynomial machinery	232
51.6	The four-element field $GF(4)$ via Kronecker's construction . . .	233
51.7	The eight-element field $GF(8)$ via Kronecker's construction . . .	234
51.8	The nine-element field $GF(9)$ via Kronecker's construction . . .	235
52	Divisibility in commutative rings: units, associates, irreducibles, primes	235
52.1	Divisibility and units	235
52.2	Associated elements	237
52.3	Greatest common divisors and least common multiples	237
52.4	Irreducible and prime elements	238
52.5	In an integral domain, prime elements are irreducible	239
52.6	Divisibility via principal ideals	240
52.7	Noetherian rings	240
52.8	Principal ideal domains	241
52.8.1	The ascending chain condition on principal ideals	241
52.8.2	Bezout: existence of greatest common divisors	242
52.9	Euclidean domains	243
52.10	Factorizations into irreducibles	243
52.11	Uniqueness of factorization	244
52.12	Factorial domains (unique factorization domains)	245
52.12.1	Existence of factorizations	245
52.13	Euclidean type classes as locale Euclidean domains; the integers .	246
52.14	The general type-class bridge	246
52.15	The integers	247
53	The prime subfield of a finite integral domain	247
53.1	Finite integral domains	247
54	Modules over a ring	249
54.1	Elementary closure and arithmetic	249
54.2	Submodules	250
54.3	The sum of two submodules	251
54.4	Linear combinations	252
54.5	A vector space is a module over a field	252
55	Generators, independence, bases and free modules	252
55.1	Coefficient functions	253
55.2	Membership in the span	254
55.3	Generating sets, independence, bases	256

55.4 Uniqueness of coordinates	257
55.5 Coordinates over a basis	258
56 Finite-dimensional vector spaces over a field	259
57 The Steinitz exchange lemma and dimension over an arbitrary field	263
57.1 Span basics	263
57.2 Extension of a linear combination by zero coefficients	264
57.3 Span is a subspace	264
57.4 Span closure, idempotence and transitivity	264
57.5 Vector subtraction inside a span	266
57.6 The exchange step	266
57.7 Independence as irredundancy	266
57.8 Bases span and are independent	267
57.9 The Steinitz exchange lemma	268
57.10 The cardinality bound and uniqueness of dimension	268
57.11 Existence of a basis	268
58 A finite subgroup of the multiplicative group of a field is cyclic	271
58.1 The order of a power in a finite group	271
58.2 Counting elements of each order in a cyclic group	272
58.3 Counting the Residues Coprime to a Modulus	272
58.4 The abstract group criterion for being cyclic	273
58.5 The multiplicative group of a field and the root bound	273
58.6 The polynomial root bound, via the polynomial ring	274
58.7 The main theorem	275
59 A field extension as a vector space over a subfield	275
59.1 A subfield is a locale-based field	276
59.2 The extension as a vector space	276
60 Frobenius on finite subfields	278
60.1 Cardinality of finite fields	278
61 The lattice of ideals	281
61.1 Intersections of ideals	281
61.2 The ideal generated by a set	281
61.3 The product of two ideals	282
61.4 The complete lattice of ideals	283

62	Extension and contraction of ideals along a ring homomorphism	283
62.1	Contraction: the preimage of an ideal	284
62.2	Extension: the ideal generated by the image	284
62.3	The adjunction	285
63	The direct product of an indexed family of rings	287
63.1	The product locale	287
63.2	Carrier and operations	287
63.3	The additive abelian group	288
63.4	The multiplicative monoid	289
63.5	The product is a ring	289
63.6	Projections are ring homomorphisms	289
63.7	The product of a family of commutative rings is commutative . .	289
64	The Chinese Remainder Theorem for commutative rings	290
64.1	Associative-commutative rewriting in an abelian group	290
64.2	The binary direct product of two rings	290
64.3	Sum of two ideals	291
64.4	Two comaximal ideals	293
64.5	The Chinese Remainder Theorem	294
65	The n-ary Chinese Remainder Theorem	295
65.1	Pairwise-comaximal ideals	295
65.2	The canonical projection into the indexed product	296
65.3	Kernel and surjectivity	298
65.4	The n-ary Chinese Remainder Theorem	299
66	The field of fractions of an integral domain	299
66.1	Fractions and their equivalence	299
66.2	The carrier and class equality	300
66.3	The operations on classes	301
66.3.1	Representative selection	302
66.3.2	Congruence: well-definedness under the equivalence . . .	302
66.3.3	The <i>Class</i> -formulas	303
66.3.4	Closure under the operations	303
66.4	The field of fractions	304
66.4.1	The additive abelian group of <i>Fractions</i>	304
66.4.2	The multiplicative monoid of <i>Fractions</i>	304
66.4.3	Distributivity and the ring axiom	304

66.5	The canonical embedding $R \rightarrow \text{Fractions}$	305
67	The polynomial ring over a field is a Euclidean domain	306
67.1	Units and irreducibility across the seam	307
67.2	The field of remainders modulo an irreducible polynomial	308
68	Polynomials in an Indexed Family of Indeterminates	309
68.1	Submultisets	310
68.2	Carrier and coefficients	310
68.3	Operations	311
68.4	Closure under the operations	311
68.5	Additive structure	312
68.6	Multiplicative unit	312
68.7	Distributivity	312
68.8	Associativity of the convolution	313
68.9	The indexed polynomial ring	313
68.10	Constants and indeterminates	313
68.11	Indeterminates a polynomial does not involve	314
68.12	Monomials	315
68.13	Reading a polynomial off its value at an indeterminate	316
69	Polynomials over a Commutative Ring	317
70	The Univariate Case, as a Check on the Convolution	317
71	Extensions of a Field inside the Indexed Polynomial Ring	318
71.1	Operations defined only on their carrier	319
71.2	Lifted polynomials	319
71.3	What it is to be an extension	320
71.4	The ordering by inclusion	322
71.5	Powers and evaluation transfer along the ordering	323
71.6	The base extension	323
71.7	Unions of chains	324
71.8	Evaluation is the same in a member and in the union	326
71.9	The union of a nonempty chain is an extension	327
72	Transporting a Field Structure along an Injection	327
72.1	The base extension is an extension	328
72.2	A maximal extension exists	328
72.3	A fresh index always exists	328

72.4	Transporting a field onto its image, with restricted operations . .	328
73	Working Inside a Fixed Extension	329
73.1	Every polynomial over the base field splits in a maximal extension	334
74	An Extension in which Every Base Polynomial Splits	334
75	Linking the carrier-set polynomial ring to Isabelle's polynomial type	335
75.1	Finite composites are sums and products	335
75.2	The two representations of a polynomial	335
75.3	The dictionary at a type-class field	336
75.4	The ring operations correspond	337
75.5	Constants, the indeterminate, degree and evaluation	337
75.6	Splitting into linear factors	338
75.7	The dictionary read backwards	339
76	Degree of a simple algebraic extension	339
76.1	Uniqueness of the minimal polynomial	339
76.2	The minimal polynomial as a function	340
76.3	The extension degree	341
76.4	The power basis of a simple algebraic extension	341
76.5	The tower law, for a two-step simple tower	342
77	Finite field extensions and their degree	343
77.1	Finite intermediate extensions	345
78	Finite-dimensional extensions and transitivity of algebraicity	346
78.1	A finite-dimensional extension is algebraic	346
78.2	Transitivity of algebraicity	346
78.3	Algebraicity along a radical/simple tower	347
79	Generic properties of algebraic field extensions	348
80	The tower law for finite field extensions	350
81	Extensions generated by finitely many algebraic elements	352
81.1	A subfield is a finite extension of itself	353
81.2	A simple algebraic extension is finite	353
81.3	Finite extensions compose	353
81.4	Finitely many algebraic elements generate a finite extension . . .	353

81.5	Coordinates over a finite extension	353
82	Homogeneous linear systems with more unknowns than equations	354
83	The Algebraic Closure of a Field	355
83.1	An extension of the whole type in which every polynomial splits	355
83.2	The extension as a type	356
83.3	The base field inside the extension	359
83.4	Every polynomial over the base subfield splits	360
83.5	The algebraic closure	361
83.6	The closure is algebraically closed	362
84	Module homomorphisms	363
84.1	Basic properties	364
84.2	Composition and inverses	364
84.3	The kernel	365
84.4	The image	365
84.5	Injectivity criterion	366
85	Quotient modules and the first isomorphism theorem	366
85.1	The quotient module of a submodule	366
85.1.1	The induced scaling is well-defined	367
85.1.2	The quotient scaling operation and the quotient module	368
85.1.3	The natural projection	368
86	The first isomorphism theorem for modules	369
86.1	The kernel-quotient and the induced map	369
86.2	The image $\eta^{-1} M$ is a submodule of the target	370
86.3	The induced map is a module homomorphism, injective, surjective	370
87	The second and third isomorphism theorems for modules	371
87.1	The kernel of the projection is N/K	372
87.2	The third isomorphism theorem	373
87.3	The second isomorphism theorem	374
88	Complementary submodules and projections	376
89	Exact sequences of modules	378
90	The direct product and direct sum of a family of modules	381

90.1	The product locale	382
90.2	Carrier and operations	382
90.3	The additive abelian group of the product	383
90.4	The product is a module	383
90.5	Projections	383
90.6	The direct sum	384
91	The universal property of a free module	384
91.1	The twisted sum	385
91.2	The extension is well defined	386
91.3	The extension	386
92	The rank–nullity theorem	387
92.1	Finite-dimensional consequences	389
93	Isomorphisms of finite-dimensional vector spaces	390
94	Complements in finite-dimensional vector spaces	392
94.1	Dimension of quotient spaces	392
94.2	Grassmann’s dimension formula	393
94.3	Complements in finite-dimensional vector spaces	396
95	Simple Algebraic Extensions inside the Complex Numbers	397
95.1	Evaluation of a rational polynomial at a complex point	397
95.2	The kernel is generated by the minimal polynomial	398
95.3	The simple extension $\mathbb{Q}(a)$ and the base isomorphism	398
96	The Galois correspondence between intermediate fields and subgroups	399
96.1	The fixed field of a set of automorphisms	400
96.2	Intermediate fields and the relative Galois group	400
96.3	The adjunction	401
96.4	The Galois connection	402
97	Normal and separable complex field extensions	404
98	Fixed fields of finite normal and separable extensions	406
99	Primitive elements of finite field extensions	407
99.1	The two-generator theorem over an infinite base	407
99.2	Finite separable extensions are simple	408

100	The order of the Galois group of a finite normal separable extension	408
100.1	The minimal polynomial is irreducible over the base	409
100.2	An F -automorphism of $F(a)$ is determined by its value at a . . .	409
100.3	The order of the Galois group of a simple normal extension . . .	409
100.4	Separability in characteristic 0 : the number of conjugates is the degree	410
100.5	The headline: order of the Galois group of a simple normal extension	410
100.6	Generic finite normal and separable extensions	411
101	Field-side inverse and degree formulas for simple normal extensions	412
101.1	Algebraicity of generated splitting fields	412
102	Generic restriction of field automorphisms	413
102.1	The inductive step: solvability climbs the tower	415
102.2	Complex splitting-field extension of generic restriction	415
102.3	A normal solvable tower has a solvable Galois group	416
102.3.1	The trivial Galois group	416
102.3.2	Normal solvable towers	416
102.3.3	The assembly	416
103	Restriction along a stable intermediate field of a splitting field	417
104	Artin's lemma: a finite automorphism group bounds the degree over its fixed field	419
104.1	Artin's lemma	419
104.2	Transporting a dependence along an automorphism	420
104.3	Descending the coefficients to the fixed field	421
104.4	Dedekind's lemma: distinct automorphisms are linearly independent	422
105	Artin's degree bound: the fixed field of a finite group has small index	423
105.1	The fixed field sits below K as a tower	423
105.2	Artin's bound	423
105.3	The reverse bound: the Galois group is no larger than the degree	425
105.4	Artin's theorem	426
106	The fundamental theorem for finite Galois extensions	427

106.1	The finite correspondence	428
106.2	Degree and index formulas	429
106.3	Normal intermediate fields	430
106.4	Restriction and the quotient theorem	430
107	Radical extensions and solvability by radicals	431
107.1	Radical steps and towers	432
107.2	Solvability by radicals	432
107.3	Concatenation of radical towers	433
107.4	Base change: lifting a radical tower to a larger base	433
108	Stability of root-generated intermediate fields	434
108.1	An automorphism fixes the base field setwise	434
108.2	Root-generated fields are stable	434
108.3	A field homomorphism carries radical steps to radical steps . . .	435
108.4	A field homomorphism carries radical towers to radical towers . .	435
108.5	The top of a radical tower is a subfield containing the base . . .	436
108.6	Splicing a list of towers over a common base into one radical tower	436
108.7	A radical tower is increasing; every field lies in the top	437
109	Finite field extensions	437
110	Galois groups of finite fields	439
111	The New-Algebra Development	442


```

theory Set-Theory
  imports HOL-Library.FuncSet HOL-Library.Disjoint-Sets

begin

no-notation divide (infixl <'/'> 70)
no-notation inverse-divide (infixl <'/'> 70)
no-notation quotient (infixl <'/'> 90)

abbreviation identity  $S \equiv (\lambda x \in S. x)$ 

theorem bij-betw-iff-has-inverse:
  assumes  $\alpha \in S \rightarrow_E T$ 
  shows  $\text{bij-betw } \alpha \ S \ T \longleftrightarrow (\exists \beta \in T \rightarrow_E S. \text{compose } S \ \beta \ \alpha = \text{identity } S \wedge \text{compose } T \ \alpha \ \beta = \text{identity } T)$ 
  (is  $\longleftrightarrow (\exists \beta \in T \rightarrow_E S. ?INV \ \beta)$ )
  <proof>

```

0.1 Compatibility with Ballarin's original development

Maps as extensional HOL functions

p 5, ll 21–25

```

locale map =
  fixes  $\alpha$  and  $S$  and  $T$ 
  assumes graph [intro, simp]:  $\alpha \in S \rightarrow_E T$ 
begin

```

p 5, ll 21–25

```

lemma map-closed [intro, simp]:
   $a \in S \implies \alpha \ a \in T$ 
  <proof>

```

p 5, ll 21–25

```

lemma map-undefined [intro]:
   $a \notin S \implies \alpha \ a = \text{undefined}$ 
  <proof>

```

end

p 7, ll 7–8

```

locale surjective-map = map +
  assumes surjective [intro]:  $\alpha \ ' S = T$ 

```

p 7, ll 8–9

```

locale injective-map = map +

```

```

assumes injective [intro, simp]: inj-on  $\alpha$   $S$ 

locale bijjective-map = map +
  assumes bijjective [intro, simp]: bij-betw  $\alpha$   $S$   $T$ 

begin

p 7, ll 9–10
sublocale surjective-map  $\langle$ proof $\rangle$ 

p 7, ll 9–10
sublocale injective-map  $\langle$ proof $\rangle$ 

p 9, ll 12–13
sublocale inverse: map restrict (inv-into S  $\alpha$ ) T T S
 $\langle$ proof $\rangle$ 

p 9, ll 12–13
lemma bij-betw-inverse: bij-betw (restrict (inv-into S  $\alpha$ ) T) T S
 $\langle$ proof $\rangle$ 

end


locale Equivalence =
  fixes  $S$  and  $E$ 
  assumes eqv: equiv  $S$   $E$ 
begin

lemma closed [intro, simp]:  $E \subseteq S \times S$ 
 $\langle$ proof $\rangle$ 

lemma reflexive [intro, simp]:  $a \in S \implies (a, a) \in E$ 
 $\langle$ proof $\rangle$ 

lemma symmetric [sym]:  $(a, b) \in E \implies (b, a) \in E$ 
 $\langle$ proof $\rangle$ 

lemma transitive [trans]:  $\llbracket (a, b) \in E; (b, c) \in E \rrbracket \implies (a, c) \in E$ 
 $\langle$ proof $\rangle$ 

lemma left-closed [intro]:
   $(a, b) \in E \implies a \in S$ 
 $\langle$ proof $\rangle$ 

p 11, ll 6–11
lemma right-closed [intro]:
   $(a, b) \in E \implies b \in S$ 

```

$\langle proof \rangle$

Equivalence classes

definition $Class \equiv (\lambda a \in S. E''\{a\})$

p 11, ll 24–26

lemma *Class-closed* [*dest*]:
 $\llbracket a \in Class\ b; b \in S \rrbracket \implies a \in S$
 $\langle proof \rangle$

p 11, ll 24–26

lemma *Class-closed2* [*intro, simp*]:
 $a \in S \implies Class\ a \subseteq S$
 $\langle proof \rangle$

p 11, ll 24–26

lemma *Class-undefined* [*intro, simp*]:
 $a \notin S \implies Class\ a = undefined$
 $\langle proof \rangle$

p 11, ll 24–26

lemma *ClassI* [*intro, simp*]:
 $(a, b) \in E \implies a \in Class\ b$
 $\langle proof \rangle$

p 11, ll 24–26

lemma *Class-revI* [*intro, simp*]:
 $(a, b) \in E \implies b \in Class\ a$
 $\langle proof \rangle$

p 11, ll 24–26

lemma *ClassD* [*dest*]:
 $\llbracket b \in Class\ a; a \in S \rrbracket \implies (b, a) \in E$
 $\langle proof \rangle$

p 11, ll 30–31

theorem *Class-self* [*intro, simp*]:
 $a \in S \implies a \in Class\ a$
 $\langle proof \rangle$

p 11, l 31; p 12, l 1

theorem *Class-Union* [*simp*]:
 $(\bigcup_{a \in S. Class\ a}) = S$
 $\langle proof \rangle$

p 11, ll 2–3

theorem *Class-subset*:

$(a, b) \in E \implies \text{Class } a \subseteq \text{Class } b$
 $\langle \text{proof} \rangle$

p 11, ll 3–4

theorem *Class-eq:*

$(a, b) \in E \implies \text{Class } a = \text{Class } b$
 $\langle \text{proof} \rangle$

p 12, ll 1–5

theorem *Class-equivalence:*

$\llbracket a \in S; b \in S \rrbracket \implies \text{Class } a = \text{Class } b \longleftrightarrow (a, b) \in E$
 $\langle \text{proof} \rangle$

p 12, ll 5–7

theorem *not-disjoint-implies-equal:*

assumes $\text{Class } a \cap \text{Class } b \neq \{\}$
assumes $a \in S \ b \in S$
shows $\text{Class } a = \text{Class } b$
 $\langle \text{proof} \rangle$

definition *Partition = Class ‘ S*

notation *Equivalence.Partition* (**infixl** $\langle ' / \rangle$ 75)

p 12, ll 7–8

lemma *Class-in-Partition* [*intro, simp*]:

$a \in S \implies \text{Class } a \in \text{Partition}$
 $\langle \text{proof} \rangle$

p 12, ll 7–8

theorem *partition:*

partition-on S Partition
 $\langle \text{proof} \rangle$

end

context *Equivalence*

begin

p 12, ll 18–20

lemma *representant-exists:* $A \in S / E \implies \exists a \in S. a \in A \wedge A = \text{Class } a$
 $\langle \text{proof} \rangle$

p 12, ll 18–20

lemma *quotient-ClassE:* $A \in S / E \implies (\bigwedge a. a \in S \implies P (\text{Class } a)) \implies P A$
 $\langle \text{proof} \rangle$

end

lemma *equivalenceI*:

assumes *closed*: $E \subseteq S \times S$
and *reflexive*: $\bigwedge a. a \in S \implies (a, a) \in E$
and *symmetric*: $\bigwedge a b. (a, b) \in E \implies (b, a) \in E$
and *transitive*: $\bigwedge a b c. [(a, b) \in E; (b, c) \in E] \implies (a, c) \in E$
shows *Equivalence* $S E$
 $\langle proof \rangle$

p 12, ll 21–23

sublocale *Equivalence* $\subseteq$ *natural*: *surjective-map* *Class* $S S / E$
 $\langle proof \rangle$

Technical device to achieve Jacobson’s syntax; context where α is not a parameter.

p 12, ll 25–26

locale *fiber-relation-notation* = **fixes** $S :: 'a \text{ set}$ **begin**

p 12, ll 25–26

definition *Fiber-Relation* $(\langle E'(-) \rangle)$ **where** *Fiber-Relation* $\alpha = \{(x, y). x \in S \wedge y \in S \wedge \alpha x = \alpha y\}$

end

Context where classes and the induced map are defined through the fiber relation. This will be the case for monoid homomorphisms but not group homomorphisms.

Avoids infinite interpretation chain.

p 12, ll 25–26

locale *fiber-relation* = *map* **begin**

Install syntax

p 12, ll 25–26

sublocale *fiber-relation-notation* $\langle proof \rangle$

p 12, ll 26–27

sublocale *Equivalence* **where** $E = E(\alpha)$
 $\langle proof \rangle$

“define $\bar{\alpha}$ by $\bar{\alpha}(\bar{a}) = \alpha(a)$ ”

p 13, ll 8–9

definition *induced* = $(\lambda A \in S / E(\alpha). \text{THE } b. \exists a \in A. b = \alpha a)$

p 13, l 10

theorem *Fiber-equality*:

$\llbracket a \in S; b \in S \rrbracket \implies \text{Class } a = \text{Class } b \longleftrightarrow \alpha a = \alpha b$
 $\langle \text{proof} \rangle$

p 13, ll 8–9

theorem *induced-Fiber-simp* [simp]:

assumes [intro, simp]: $a \in S$
shows *induced* ($\text{Class } a$) = αa
 $\langle \text{proof} \rangle$

p 13, ll 10–11

interpretation *induced*: *map induced* $S / E(\alpha) T$
 $\langle \text{proof} \rangle$

p 13, ll 12–13

sublocale *induced*: *injective-map induced* $S / E(\alpha) T$
 $\langle \text{proof} \rangle$

p 13, ll 16–19

theorem *factorization-lemma*:

$a \in S \implies \text{compose } S \text{ induced } \text{Class } a = \alpha a$
 $\langle \text{proof} \rangle$

p 13, ll 16–19

theorem *factorization* [simp]: *compose* S *induced* $\text{Class} = \alpha$
 $\langle \text{proof} \rangle$

p 14, ll 2–4

theorem *uniqueness*:

assumes *map*: $\beta \in S / E(\alpha) \rightarrow_E T$
and *factorization*: *compose* $S \beta \text{Class} = \alpha$
shows $\beta = \text{induced}$
 $\langle \text{proof} \rangle$

end

hide-const (**open**) *map*

end

theory *Group-Theory*

imports *Set-Theory*

begin

1 Monoids and Groups

1.1 Monoids of Transformations and Abstract Monoids

```

locale Monoid =
  fixes  $M$  and composition (infixl  $\langle \cdot \rangle$  70) and unit ( $\langle 1 \rangle$ )
  assumes composition-closed [intro, simp]:  $\llbracket a \in M; b \in M \rrbracket \implies a \cdot b \in M$ 
  and unit-closed [intro, simp]:  $1 \in M$ 
  and associative [intro]:  $\llbracket a \in M; b \in M; c \in M \rrbracket \implies (a \cdot b) \cdot c = a \cdot (b \cdot c)$ 
  and left-unit [intro, simp]:  $a \in M \implies 1 \cdot a = a$ 
  and right-unit [intro, simp]:  $a \in M \implies a \cdot 1 = a$ 

```

p 29, ll 27–28

```

locale Submonoid = Monoid  $M$  ( $\cdot$ ) 1
  for  $N$  and  $M$  and composition (infixl  $\langle \cdot \rangle$  70) and unit ( $\langle 1 \rangle$ ) +
  assumes subset:  $N \subseteq M$ 
  and sub-composition-closed:  $\llbracket a \in N; b \in N \rrbracket \implies a \cdot b \in N$ 
  and sub-unit-closed:  $1 \in N$ 
begin

```

p 29, ll 27–28

```

lemma sub [intro, simp]:
   $a \in N \implies a \in M$ 
   $\langle \text{proof} \rangle$ 

```

p 29, ll 32–33

```

sublocale sub: Monoid  $N$  ( $\cdot$ ) 1
   $\langle \text{proof} \rangle$ 

```

end

p 29, ll 33–34

```

theorem submonoid-transitive:
  assumes Submonoid  $K$   $N$  composition unit
  and Submonoid  $N$   $M$  composition unit
  shows Submonoid  $K$   $M$  composition unit
   $\langle \text{proof} \rangle$ 

```

p 28, l 23

```

locale transformations =
  fixes  $S :: 'a$  set

```

Monoid of all transformations

p 28, ll 23–24

```

sublocale transformations  $\subseteq$  Monoid  $S \rightarrow_E S$  compose identity  $S$ 
   $\langle \text{proof} \rangle$ 

```

N is a Monoid of transformations of the set S .

p 29, ll 34-36

locale *transformation-monoid* =
 transformations $S + \text{Submonoid } M \rightarrow_E S$ *compose* S *identity* S **for** M **and** S
begin

p 29, ll 34-36

lemma *transformation-closed* [*intro*, *simp*]:
 $\llbracket \alpha \in M; x \in S \rrbracket \implies \alpha \ x \in S$
 $\langle \text{proof} \rangle$

p 29, ll 34-36

lemma *transformation-undefined* [*intro*, *simp*]:
 $\llbracket \alpha \in M; x \notin S \rrbracket \implies \alpha \ x = \text{undefined}$
 $\langle \text{proof} \rangle$

end

1.2 Groups of Transformations and Abstract Groups

context *Monoid* **begin**

Invertible elements

p 31, ll 3-5

definition *invertible* **where** *invertible* $u \equiv (\exists v \in M. u \cdot v = \mathbf{1} \wedge v \cdot u = \mathbf{1})$

p 31, ll 3-5

lemma *invertibleI* [*intro*]:
 $\llbracket u \cdot v = \mathbf{1}; v \cdot u = \mathbf{1}; v \in M \rrbracket \implies \text{invertible } u$
 $\langle \text{proof} \rangle$

p 31, ll 3-5

lemma *invertibleE* [*elim*]:
 $\llbracket \text{invertible } u; \bigwedge v. \llbracket u \cdot v = \mathbf{1} \wedge v \cdot u = \mathbf{1}; v \in M \rrbracket \implies P \rrbracket \implies P$
 $\langle \text{proof} \rangle$

p 31, ll 6-7

theorem *inverse-unique*:
 $\llbracket u \cdot v' = \mathbf{1}; v \cdot u = \mathbf{1}; u \in M; v \in M; v' \in M \rrbracket \implies v = v'$
 $\langle \text{proof} \rangle$

p 31, l 7

definition *inverse* **where** *inverse* $\equiv (\lambda u \in M. \text{THE } v. v \in M \wedge u \cdot v = \mathbf{1} \wedge v \cdot u = \mathbf{1})$

p 31, l 7

theorem *inverse-equality*:

$\llbracket u \cdot v = \mathbf{1}; v \cdot u = \mathbf{1}; u \in M; v \in M \rrbracket \implies \text{inverse } u = v$
 $\langle \text{proof} \rangle$

p 31, l 7

lemma *invertible-inverse-closed* [intro, simp]:
 $\llbracket \text{invertible } u; u \in M \rrbracket \implies \text{inverse } u \in M$
 $\langle \text{proof} \rangle$

p 31, l 7

lemma *inverse-undefined* [intro, simp]:
 $u \notin M \implies \text{inverse } u = \text{undefined}$
 $\langle \text{proof} \rangle$

p 31, l 7

lemma *invertible-left-inverse* [simp]:
 $\llbracket \text{invertible } u; u \in M \rrbracket \implies \text{inverse } u \cdot u = \mathbf{1}$
 $\langle \text{proof} \rangle$

p 31, l 7

lemma *invertible-right-inverse* [simp]:
 $\llbracket \text{invertible } u; u \in M \rrbracket \implies u \cdot \text{inverse } u = \mathbf{1}$
 $\langle \text{proof} \rangle$

p 31, l 7

lemma *invertible-left-cancel* [simp]:
 $\llbracket \text{invertible } x; x \in M; y \in M; z \in M \rrbracket \implies x \cdot y = x \cdot z \longleftrightarrow y = z$
 $\langle \text{proof} \rangle$

p 31, l 7

lemma *invertible-right-cancel* [simp]:
 $\llbracket \text{invertible } x; x \in M; y \in M; z \in M \rrbracket \implies y \cdot x = z \cdot x \longleftrightarrow y = z$
 $\langle \text{proof} \rangle$

p 31, l 7

lemma *inverse-unit* [simp]: $\text{inverse } \mathbf{1} = \mathbf{1}$
 $\langle \text{proof} \rangle$

p 31, ll 7–8

theorem *invertible-inverse-invertible* [intro, simp]:
 $\llbracket \text{invertible } u; u \in M \rrbracket \implies \text{invertible } (\text{inverse } u)$
 $\langle \text{proof} \rangle$

p 31, l 8

theorem *invertible-inverse-inverse* [simp]:
 $\llbracket \text{invertible } u; u \in M \rrbracket \implies \text{inverse } (\text{inverse } u) = u$
 $\langle \text{proof} \rangle$

end

context *Submonoid* **begin**

Reasoning about *invertible* and *local.inverse* in submonoids.

p 31, l 7

lemma *submonoid-invertible* [*intro*, *simp*]:
 $\llbracket \text{sub.invertible } u; u \in N \rrbracket \implies \text{invertible } u$
 ⟨*proof*⟩

p 31, l 7

lemma *submonoid-inverse-closed* [*intro*, *simp*]:
 $\llbracket \text{sub.invertible } u; u \in N \rrbracket \implies \text{inverse } u \in N$
 ⟨*proof*⟩

end

Def 1.2

p 31, ll 9–10

locale *Group* =

Monoid G ($\cdot$) **1** **for** G **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle 1 \rangle$) +
assumes *invertible* [*simp*, *intro*]: $u \in G \implies \text{invertible } u$

A self-contained introduction rule for groups: the monoid laws together with the existence of a two-sided inverse for every element (stated directly, without reference to the *Monoid.invertible* predicate, which would otherwise require a separate monoid interpretation).

lemma *GroupI*:

fixes G **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle 1 \rangle$)
assumes $\bigwedge a b. \llbracket a \in G; b \in G \rrbracket \implies a \cdot b \in G$
and $1 \in G$
and $\bigwedge a b c. \llbracket a \in G; b \in G; c \in G \rrbracket \implies (a \cdot b) \cdot c = a \cdot (b \cdot c)$
and $\bigwedge a. a \in G \implies 1 \cdot a = a$
and $\bigwedge a. a \in G \implies a \cdot 1 = a$
and *inv*: $\bigwedge u. u \in G \implies \exists v \in G. u \cdot v = 1 \wedge v \cdot u = 1$
shows *Group* G ($\cdot$) **1**

⟨*proof*⟩

p 31, ll 11–12

The first parameter H is the *subgroup* and G the ambient group, mirroring the parameter order of *Submonoid*. At all use sites one writes *subgroup* H G (subgroup then ambient); the later locales *subgroup-of-group* and *normal-subgroup* take their parameters in the same order.

locale *Subgroup* = *Submonoid* H G ($\cdot$) **1** + *sub*: *Group* H ($\cdot$) **1**
for H **and** G **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle 1 \rangle$)

begin

Reasoning about *invertible* and *local.inverse* in subgroups.

p 31, ll 11–12

lemma *subgroup-inverse-equality* [simp]:
 $u \in H \implies \text{inverse } u = \text{sub.inverse } u$
 ⟨proof⟩

p 31, ll 11–12

lemma *subgroup-inverse-iff* [simp]:
 $\llbracket \text{invertible } x; x \in G \rrbracket \implies \text{inverse } x \in H \longleftrightarrow x \in H$
 ⟨proof⟩

end

A subgroup is in particular a group in its own right.

lemma *subgroup-imp-Group*:
 assumes *Subgroup H M composition unit*
 shows *Group H composition unit*
 ⟨proof⟩

p 31, ll 11–12

lemma *subgroup-transitive* [trans]:
 assumes *Subgroup K H composition unit*
 and *Subgroup H G composition unit*
 shows *Subgroup K G composition unit*
 ⟨proof⟩

lemma *subgroup-restrict*:
 assumes *I: Subgroup I G composition unit*
 and *H: Subgroup H G composition unit*
 and $I \subseteq H$
 shows *Subgroup I H composition unit*
 ⟨proof⟩

context *Monoid* **begin**

Jacobson states both directions, but the other one is trivial.

p 31, ll 12–15

theorem *subgroupI*:
 fixes *G*
 assumes *subset [THEN subsetD, intro]: G ⊆ M*
 and [intro]: $1 \in G$
 and [intro]: $\bigwedge g h. \llbracket g \in G; h \in G \rrbracket \implies g \cdot h \in G$
 and [intro]: $\bigwedge g. g \in G \implies \text{invertible } g$
 and [intro]: $\bigwedge g. g \in G \implies \text{inverse } g \in G$
 shows *Subgroup G M (·) 1*

$\langle proof \rangle$

p 31, l 16

definition $Units = \{u \in M. \text{invertible } u\}$

p 31, l 16

lemma *mem-UnitsI*:

$\llbracket \text{invertible } u; u \in M \rrbracket \implies u \in Units$

$\langle proof \rangle$

p 31, l 16

lemma *mem-UnitsD*:

$\llbracket u \in Units \rrbracket \implies \text{invertible } u \wedge u \in M$

$\langle proof \rangle$

p 31, ll 16–21

interpretation *units*: *Subgroup Units M*

$\langle proof \rangle$

p 31, ll 21–22

theorem *group-of-Units* [*intro*, *simp*]:

Group Units $(\cdot) \mathbf{1}$

$\langle proof \rangle$

p 31, l 19

lemma *composition-invertible* [*simp*, *intro*]:

$\llbracket \text{invertible } x; \text{invertible } y; x \in M; y \in M \rrbracket \implies \text{invertible } (x \cdot y)$

$\langle proof \rangle$

p 31, l 20

lemma *unit-invertible*:

invertible $\mathbf{1}$

$\langle proof \rangle$

Useful simplification rules

p 31, l 22

lemma *invertible-right-inverse2*:

$\llbracket \text{invertible } u; u \in M; v \in M \rrbracket \implies u \cdot (\text{inverse } u \cdot v) = v$

$\langle proof \rangle$

p 31, l 22

lemma *invertible-left-inverse2*:

$\llbracket \text{invertible } u; u \in M; v \in M \rrbracket \implies \text{inverse } u \cdot (u \cdot v) = v$

$\langle proof \rangle$

p 31, l 22

lemma *inverse-composition-commute*:
assumes *[simp]: invertible x invertible y x ∈ M y ∈ M*
shows *inverse (x · y) = inverse y · inverse x*
 $\langle proof \rangle$

lemma *conjugate-composition*:
assumes *invertible g g ∈ M a ∈ M b ∈ M*
shows *inverse g · (a · b) · g =*
(inverse g · a · g) · (inverse g · b · g)
 $\langle proof \rangle$

end

context *Group* **begin**

The whole group is a subgroup of itself.

lemma *group-self-subgroup: Subgroup G G (·) 1*
 $\langle proof \rangle$

The trivial subgroup, consisting of the unit alone.

lemma *trivial-subgroup: Subgroup {1} G (·) 1*
 $\langle proof \rangle$

1.3 The subgroup generated by a set

The subgroup generated by a set S of elements of a group, defined inductively as the smallest set containing $S \cap G$ and closed under the group operations. This basic construction underlies commutators, the derived series, and solvability.

inductive-set *generate :: 'a set $\Rightarrow$ 'a set for S :: 'a set*
where
unit: 1 ∈ generate S
| incl: x ∈ S ∩ G $\Rightarrow$ x ∈ generate S
| comp: [x ∈ generate S; y ∈ generate S] $\Rightarrow$ x · y ∈ generate S
| inv: x ∈ generate S $\Rightarrow$ inverse x ∈ generate S

definition *Gen :: 'a set $\Rightarrow$ 'a set ($\langle \cdot \rangle$)*
where *$\langle S \rangle \equiv generate (S \cap G)$*

Everything generated lies in the carrier.

lemma *generate-subset-carrier: generate S $\subseteq$ G*
 $\langle proof \rangle$

lemma *Gen-subset-carrier: $\langle S \rangle \subseteq G$*
 $\langle proof \rangle$

The generated set is a subgroup.

lemma *generate-is-subgroup: Subgroup (generate S) G (·) 1*
 $\langle proof \rangle$

lemma *Gen-is-subgroup*: Subgroup $\langle S \rangle \ G \ (\cdot) \ 1$
 $\langle proof \rangle$

The generating set is included (intersected with the carrier).

lemma *generate-incl*: $S \cap G \subseteq generate\ S$
 $\langle proof \rangle$

lemma *Gen-incl*: $S \cap G \subseteq \langle S \rangle$
 $\langle proof \rangle$

generate S is the smallest subgroup containing $S \cap G$.

lemma *generate-minimal*:
assumes Subgroup $H \ G \ (\cdot) \ 1$ **and** $S \cap G \subseteq H$
shows $generate\ S \subseteq H$
 $\langle proof \rangle$

lemma *Gen-minimal*:
assumes Subgroup $H \ G \ (\cdot) \ 1$ **and** $S \cap G \subseteq H$
shows $\langle S \rangle \subseteq H$
 $\langle proof \rangle$

Monotonicity.

lemma *generate-mono*:
assumes $S \subseteq T$
shows $generate\ S \subseteq generate\ T$
 $\langle proof \rangle$

lemma *Gen-mono*:
assumes $S \subseteq T$
shows $\langle S \rangle \subseteq \langle T \rangle$
 $\langle proof \rangle$

A subgroup generates itself.

lemma *generate-subgroup-eq*:
assumes Subgroup $H \ G \ (\cdot) \ 1$ **and** $H \subseteq G$
shows $generate\ H = H$
 $\langle proof \rangle$

lemma *Gen-subgroup-eq*:
assumes Subgroup $H \ G \ (\cdot) \ 1$
shows $\langle H \rangle = H$
 $\langle proof \rangle$

Idempotence.

lemma *Gen-idem*: $\langle \langle S \rangle \rangle = \langle S \rangle$
 $\langle proof \rangle$

The intersection characterisation: $\langle S \rangle$ is the intersection of all subgroups containing $S \cap G$.

lemma *Gen-eq-Inter*:

$\langle S \rangle = \bigcap \{H. \text{Subgroup } H \ G \ (\cdot) \ \mathbf{1} \wedge S \cap G \subseteq H\}$
 $\langle proof \rangle$

The set generated inside a subgroup (with the restricted operation) is contained in the set generated in the ambient group. A locality property of *generate*.

lemma *generate-subgroup-subset*:

assumes $H: \text{Subgroup } H \ G \ (\cdot) \ \mathbf{1}$
shows $\text{Group.generate } H \ (\cdot) \ \mathbf{1} \ S \subseteq \text{generate } S$
 $\langle proof \rangle$

end

p 31, l 24

context *transformations* **begin**

p 31, ll 25–26

theorem *invertible-is-bijective*:

assumes $\text{dom}: \alpha \in S \rightarrow_E S$
shows $\text{invertible } \alpha \longleftrightarrow \text{bij-betw } \alpha \ S \ S$
 $\langle proof \rangle$

p 31, ll 26–27

theorem *Units-bijective*:

$\text{Units} = \{\alpha \in S \rightarrow_E S. \text{bij-betw } \alpha \ S \ S\}$
 $\langle proof \rangle$

p 31, ll 26–27

lemma *Units-bij-betwI* [*intro, simp*]:

$\alpha \in \text{Units} \implies \text{bij-betw } \alpha \ S \ S$
 $\langle proof \rangle$

p 31, ll 26–27

lemma *Units-bij-betwD* [*dest, simp*]:

$\llbracket \alpha \in S \rightarrow_E S; \text{bij-betw } \alpha \ S \ S \rrbracket \implies \alpha \in \text{Units}$
 $\langle proof \rangle$

p 31, ll 28–29

abbreviation $\text{Sym} \equiv \text{Units}$

p 31, ll 26–28

sublocale *symmetric*: $\text{Group } \text{Sym} \text{ compose } S \text{ identity } S$

$\langle proof \rangle$

end

p 32, ll 18–19

locale *transformation-group* =
 transformations S + *symmetric*: Subgroup G *Sym* compose S identity S **for** G **and**
 S
begin

p 32, ll 18–19

lemma *transformation-group-closed* [*intro*, *simp*]:
 $\llbracket \alpha \in G; x \in S \rrbracket \implies \alpha x \in S$
 $\langle proof \rangle$

p 32, ll 18–19

lemma *transformation-group-undefined* [*intro*, *simp*]:
 $\llbracket \alpha \in G; x \notin S \rrbracket \implies \alpha x = \text{undefined}$
 $\langle proof \rangle$

end

1.4 Isomorphisms. Cayley’s Theorem

Def 1.3

p 37, ll 7–11

locale *Monoid-isomorphism* =
 bijective-map η M M' + *source*: Monoid M $(\cdot)$ $\mathbf{1}$ + *target*: Monoid M' $(\cdot')$ $\mathbf{1}'$
 for η **and** M **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$)
 and M' **and** *composition'* (**infixl** $\langle \cdot' \rangle$ 70) **and** *unit'* ($\langle \mathbf{1}' \rangle$) +
 assumes *commutes-with-composition*: $\llbracket x \in M; y \in M \rrbracket \implies \eta x \cdot' \eta y = \eta (x \cdot y)$
 and *commutes-with-unit*: $\eta \mathbf{1} = \mathbf{1}'$

p 37, l 10

definition *isomorphic-as-monoids* (**infixl** $\langle \cong_M \rangle$ 50)
 where $\mathcal{M} \cong_M \mathcal{M}' \longleftrightarrow (\text{let } (M, \text{composition}, \text{unit}) = \mathcal{M}; (M', \text{composition}', \text{unit}') = \mathcal{M}' \text{ in}$
 $(\exists \eta. \text{Monoid-isomorphism } \eta \text{ } M \text{ composition unit } M' \text{ composition}' \text{ unit}'))$

p 37, ll 11–12

locale *Monoid-isomorphism'* =
 bijective-map η M M' + *source*: Monoid M $(\cdot)$ $\mathbf{1}$ + *target*: Monoid M' $(\cdot')$ $\mathbf{1}'$
 for η **and** M **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$)
 and M' **and** *composition'* (**infixl** $\langle \cdot' \rangle$ 70) **and** *unit'* ($\langle \mathbf{1}' \rangle$) +
 assumes *commutes-with-composition*: $\llbracket x \in M; y \in M \rrbracket \implies \eta x \cdot' \eta y = \eta (x \cdot y)$

p 37, ll 11–12

sublocale *Monoid-isomorphism* $\subseteq$ *Monoid-isomorphism'*

<proof>

Both definitions are equivalent.

p 37, ll 12–15

sublocale *Monoid-isomorphism'* $\subseteq$ *Monoid-isomorphism*
<proof>

context *Monoid-isomorphism*
begin

p 37, ll 30–33

theorem *inverse-monoid-isomorphism:*
Monoid-isomorphism (*restrict* (*inv-into* M η) M') $M' (\cdot) \mathbf{1}' M (\cdot) \mathbf{1}$
<proof>

end

We only need that η is symmetric.

p 37, ll 28–29

theorem *isomorphic-as-monoids-symmetric:*
 $(M, \text{composition}, \text{unit}) \cong_M (M', \text{composition}', \text{unit}') \implies (M', \text{composition}', \text{unit}') \cong_M (M, \text{composition}, \text{unit})$
<proof>

Reflexivity and transitivity complete the picture: isomorphism is an equivalence relation on monoid triples, which is what lets isomorphism classes be formed as a quotient.

lemma *isomorphic-as-monoids-refl:*
assumes *Monoid* M *composition* *unit*
shows $(M, \text{composition}, \text{unit}) \cong_M (M, \text{composition}, \text{unit})$
<proof>

lemma *isomorphic-as-monoids-transitive:*
assumes $(M, \text{composition}, \text{unit}) \cong_M (M', \text{composition}', \text{unit}')$
and $(M', \text{composition}', \text{unit}') \cong_M (M'', \text{composition}'', \text{unit}'')$
shows $(M, \text{composition}, \text{unit}) \cong_M (M'', \text{composition}'', \text{unit}'')$
<proof>

Being a group is a property of the isomorphism class, not of the presentation: invertibility transports along a monoid isomorphism, because the isomorphism carries a two-sided inverse to a two-sided inverse and reaches every element of the target. This is what allows isomorphism classes to be formed once, at the level of monoids, with group-ness imposed as a constraint.

lemma (*in Monoid-isomorphism*) *target-Group:*
assumes *Group* $M (\cdot) \mathbf{1}$
shows *Group* $M' (\cdot) \mathbf{1}'$

$\langle proof \rangle$

p 38, l 4

locale *left-translations-of-monoid* = *Monoid* **begin**

p 38, ll 5–7

definition *translation* ($\langle '(-)_L \rangle$) **where** *translation* = $(\lambda a \in M. \lambda x \in M. a \cdot x)$

p 38, ll 5–7

lemma *translation-map* [*intro*, *simp*]:

$a \in M \implies (a)_L \in M \rightarrow_E M$

$\langle proof \rangle$

p 38, ll 5–7

lemma *Translations-maps* [*intro*, *simp*]:

translation ‘ $M \subseteq M \rightarrow_E M$

$\langle proof \rangle$

p 38, ll 5–7

lemma *translation-apply*:

$\llbracket a \in M; b \in M \rrbracket \implies (a)_L b = a \cdot b$

$\langle proof \rangle$

p 38, ll 5–7

lemma *translation-exist*:

$f \in \text{translation} \text{ ‘ } M \implies \exists a \in M. f = (a)_L$

$\langle proof \rangle$

p 38, ll 5–7

lemmas *Translations-E* [*elim*] = *translation-exist* [*THEN* *bexE*]

p 38, l 10

theorem *translation-unit-eq* [*simp*]:

identity $M = (\mathbf{1})_L$

$\langle proof \rangle$

p 38, ll 10–11

theorem *translation-composition-eq* [*simp*]:

assumes [*simp*]: $a \in M \ b \in M$

shows *compose* $M \ (a)_L \ (b)_L = (a \cdot b)_L$

$\langle proof \rangle$

p 38, ll 7–9

sublocale *transformation*: *transformations* $M \ \langle proof \rangle$

p 38, ll 7–9

theorem *Translations-transformation-monoid:*

transformation-monoid (translation ‘ M) M

⟨proof⟩

p 38, ll 7–9

sublocale *transformation: transformation-monoid translation ‘ M M*

⟨proof⟩

p 38, l 12

lemma *translation-is-map: translation ∈ M →_E (translation ‘ M)*

⟨proof⟩

p 38, ll 12–16

theorem *translation-isomorphism [intro]:*

*Monoid-isomorphism translation M (·) **1** (translation ‘ M) (compose M) (identity M)*

⟨proof⟩

p 38, ll 12–16

sublocale *Monoid-isomorphism translation M (·) **1** translation ‘ M compose M identity M* *⟨proof⟩*

end

context *Monoid begin*

p 38, ll 1–2

interpretation *left-translations-of-monoid* *⟨proof⟩*

p 38, ll 1–2

theorem *cayley-monoid:*

*∃ M' composition' unit'. transformation-monoid M' M ∧ (M, (·), **1**) ≅_M (M', composition', unit')*

⟨proof⟩

end

p 38, l 17

locale *left-translations-of-group = Group begin*

p 38, ll 17–18

sublocale *left-translations-of-monoid where M = G* *⟨proof⟩*

p 38, ll 17–18

notation *translation (‘(-)_L’)*

The Group of left translations is a subgroup of the symmetric Group, hence *transformation.sub.invertible*.

p 38, ll 20–22

theorem *translation-invertible* [intro, simp]:
 assumes [simp]: $a \in G$
 shows *transformation.sub.invertible* $(a)_L$
 ⟨proof⟩

p 38, ll 19–20

theorem *translation-bijective* [intro, simp]:
 $a \in G \implies \text{bij-betw } (a)_L \ G \ G$
 ⟨proof⟩

p 38, ll 18–20

theorem *Translations-transformation-group*:
 transformation-group (*translation* ‘ G) G
 ⟨proof⟩

p 38, ll 18–20

sublocale *transformation: transformation-group translation* ‘ $G \ G$
 ⟨proof⟩

end

context *Group* **begin**

p 38, ll 2–3

interpretation *left-translations-of-group* ⟨proof⟩

p 38, ll 2–3

theorem *cayley-group*:
 $\exists G' \text{ composition}' \text{ unit}' . \text{transformation-group } G' \ G \wedge (G, (\cdot), \mathbf{1}) \cong_M (G', \text{composition}', \text{unit}')$
 ⟨proof⟩

end

Exercise 3

p 39, ll 9–10

locale *right-translations-of-group* = *Group* **begin**

p 39, ll 9–10

definition *translation* ($\langle '(-)_R \rangle$) **where** *translation* = $(\lambda a \in G. \lambda x \in G. x \cdot a)$

p 39, ll 9–10

abbreviation *Translations* $\equiv \text{translation}$ ‘ G

The isomorphism that will be established is a map different from *translation*.

p 39, ll 9–10

lemma *translation* $\in G \rightarrow_E \text{Translations}$
 $\langle \text{proof} \rangle$

p 39, ll 9–10

lemma *translation-map* [*intro*, *simp*]:
 $a \in G \implies (a)_R \in G \rightarrow_E G$
 $\langle \text{proof} \rangle$

p 39, ll 9–10

lemma *Translation-maps* [*intro*, *simp*]:
 $\text{Translations} \subseteq G \rightarrow_E G$
 $\langle \text{proof} \rangle$

p 39, ll 9–10

lemma *translation-apply*:
 $\llbracket a \in G; b \in G \rrbracket \implies (a)_R b = b \cdot a$
 $\langle \text{proof} \rangle$

p 39, ll 9–10

lemma *translation-exist*:
 $f \in \text{Translations} \implies \exists a \in G. f = (a)_R$
 $\langle \text{proof} \rangle$

p 39, ll 9–10

lemmas *Translations-E* [*elim*] = *translation-exist* [*THEN* *bexE*]

p 39, ll 9–10

lemma *translation-unit-eq* [*simp*]:
identity $G = (\mathbf{1})_R$
 $\langle \text{proof} \rangle$

p 39, ll 10–11

lemma *translation-composition-eq* [*simp*]:
assumes [*simp*]: $a \in G \ b \in G$
shows *compose* $G \ (a)_R \ (b)_R = (b \cdot a)_R$
 $\langle \text{proof} \rangle$

p 39, ll 10–11

sublocale *transformation*: *transformations* $G \ \langle \text{proof} \rangle$

p 39, ll 10–11

lemma *Translations-transformation-monoid*:
transformation-monoid *Translations* G
 $\langle \text{proof} \rangle$

p 39, ll 10–11

sublocale *transformation: transformation-monoid* *Translations G*
 ⟨*proof*⟩

p 39, ll 10–11

lemma *translation-invertible* [*intro, simp*]:
assumes [*simp*]: $a \in G$
shows *transformation.sub.invertible* $(a)_R$
 ⟨*proof*⟩

p 39, ll 10–11

lemma *translation-bijective* [*intro, simp*]:
 $a \in G \implies \text{bij-betw } (a)_R \ G \ G$
 ⟨*proof*⟩

p 39, ll 10–11

theorem *Translations-transformation-group*:
transformation-group *Translations G*
 ⟨*proof*⟩

p 39, ll 10–11

sublocale *transformation: transformation-group* *Translations G*
 ⟨*proof*⟩

p 39, ll 10–11

lemma *translation-inverse-eq* [*simp*]:
assumes [*simp*]: $a \in G$
shows *transformation.sub.inverse* $(a)_R = (\text{inverse } a)_R$
 ⟨*proof*⟩

p 39, ll 10–11

theorem *translation-inverse-monoid-isomorphism* [*intro*]:
Monoid-isomorphism $(\lambda a \in G. \text{transformation.symmetric.inverse } (a)_R) \ G \ (\cdot) \ \mathbf{1} \ \text{Translations}$
(compose G) (identity G)
 (is *Monoid-isomorphism ?inv - - - - -*)
 ⟨*proof*⟩

p 39, ll 10–11

sublocale *Monoid-isomorphism*
 $\lambda a \in G. \text{transformation.symmetric.inverse } (a)_R \ G \ (\cdot) \ \mathbf{1} \ \text{Translations compose G identity G}$ ⟨*proof*⟩

end

1.5 Generalized Associativity. Commutativity

p 40, l 27; p 41, ll 1–2

locale *commutative-monoid* = *Monoid* +

assumes *commutative*: $\llbracket x \in M; y \in M \rrbracket \Longrightarrow x \cdot y = y \cdot x$
begin

The left-commutativity law: together with $\llbracket ?a \in M; ?b \in M; ?c \in M \rrbracket \Longrightarrow ?a \cdot ?b \cdot ?c = ?a \cdot (?b \cdot ?c)$ and $\llbracket ?x \in M; ?y \in M \rrbracket \Longrightarrow ?x \cdot ?y = ?y \cdot ?x$ it forms a confluent rewrite system for normalising products. We bundle the three as *ac* so a proof can request full associative-commutative normalisation on demand via *simp add: ac*, rather than making them global *simp* rules (which would change the simplifier's normal form throughout the development).

lemma *left-commutative*:
assumes $x \in M \ y \in M \ z \in M$
shows $x \cdot (y \cdot z) = y \cdot (x \cdot z)$
 $\langle \text{proof} \rangle$

lemmas *ac = associative commutative left-commutative*

end

p 41, l 2

locale *Abelian-Group = Group + commutative-monoid* $G \ (\cdot) \ 1$

1.6 Orbits. Cosets of a Subgroup

context *transformation-group* **begin**

p 51, ll 18–20

definition *Orbit-Relation*
where $\text{Orbit-Relation} = \{(x, y). x \in S \wedge y \in S \wedge (\exists \alpha \in G. y = \alpha x)\}$

p 51, ll 18–20

lemma *Orbit-Relation-memI* [*intro*]:
 $\llbracket y = \alpha x; \alpha \in G; x \in S \rrbracket \Longrightarrow (x, y) \in \text{Orbit-Relation}$
 $\langle \text{proof} \rangle$

p 51, ll 18–20

lemma *Orbit-Relation-memE* [*elim*]:
 $\llbracket (x, y) \in \text{Orbit-Relation}; \bigwedge \alpha. \llbracket \alpha \in G; x \in S; y = \alpha x \rrbracket \Longrightarrow Q \rrbracket \Longrightarrow Q$
 $\langle \text{proof} \rangle$

p 51, ll 20–23, 26–27

sublocale *orbit: Equivalence* $S \ \text{Orbit-Relation}$
 $\langle \text{proof} \rangle$

p 51, ll 23–24

theorem *orbit-equality*:
 $x \in S \Longrightarrow \text{orbit.Class } x = \{\alpha x \mid \alpha. \alpha \in G\}$
 $\langle \text{proof} \rangle$

end

context *Monoid-isomorphism* **begin**

p 52, ll 16–17

theorem *image-subgroup*:

assumes *Subgroup* $G\ M\ (\cdot)\ \mathbf{1}$

shows *Subgroup* $(\eta\ \cdot\ G)\ M'\ (\cdot')\ \mathbf{1}'$

$\langle proof \rangle$

end

Technical device to achieve Jacobson’s notation for *Right-Coset* and *Left-Coset*. The definitions are pulled out of *subgroup-of-group* to a context where H is not a parameter.

p 52, l 20

locale *coset-notation* = **fixes** *composition* (**infixl** $\langle \cdot \rangle$ 70) **begin**

Equation 23

p 52, l 20

definition *Right-Coset* (**infixl** $\langle \cdot \rangle$ 70) **where** $H\ |\cdot\ x \equiv (\lambda h. h \cdot x)\ \cdot\ H$

p 53, ll 8–9

definition *Left-Coset* (**infixl** $\langle \cdot \rangle$ 70) **where** $x\ \cdot\ H \equiv (\lambda h. x \cdot h)\ \cdot\ H$

p 52, l 20

lemma *Right-Coset-memI* [*intro*]:

$h \in H \implies h \cdot x \in H\ |\cdot\ x$

$\langle proof \rangle$

p 52, l 20

lemma *Right-Coset-memE* [*elim*]:

$\llbracket a \in H\ |\cdot\ x; \bigwedge h. \llbracket h \in H; a = h \cdot x \rrbracket \implies P \rrbracket \implies P$

$\langle proof \rangle$

p 53, ll 8–9

lemma *Left-Coset-memI* [*intro*]:

$h \in H \implies x \cdot h \in x\ \cdot\ H$

$\langle proof \rangle$

p 53, ll 8–9

lemma *Left-Coset-memE* [*elim*]:

$\llbracket a \in x\ \cdot\ H; \bigwedge h. \llbracket h \in H; a = x \cdot h \rrbracket \implies P \rrbracket \implies P$

$\langle proof \rangle$

end

p 52, l 12

locale *subgroup-of-group* = *Subgroup* $H\ G\ (\cdot)\ \mathbf{1} + \text{coset-notation } (\cdot) + \text{Group } G\ (\cdot)\ \mathbf{1}$
for H **and** G **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$)
begin

p 52, ll 12–14

interpretation *left*: *left-translations-of-group* $\langle \text{proof} \rangle$

interpretation *right*: *right-translations-of-group* $\langle \text{proof} \rangle$

left.translation $\cdot\ H$ denotes Jacobson's $H_L(G)$ and *left.translation* $\cdot\ G$ denotes Jacobson's G_L .

p 52, ll 16–18

theorem *left-translations-of-subgroup-are-transformation-group* [intro]:
transformation-group (*left.translation* $\cdot\ H$) G
 $\langle \text{proof} \rangle$

p 52, l 18

interpretation *transformation-group* *left.translation* $\cdot\ H\ G$ $\langle \text{proof} \rangle$

p 52, ll 19–20

theorem *Right-Coset-is-orbit*:
 $x \in G \implies H \mid \cdot\ x = \text{orbit.Class } x$
 $\langle \text{proof} \rangle$

p 52, ll 24–25

theorem *Right-Coset-Union*:
 $(\bigcup_{x \in G} H \mid \cdot\ x) = G$
 $\langle \text{proof} \rangle$

p 52, l 26

theorem *Right-Coset-bij*:
assumes G [simp]: $x \in G\ y \in G$
shows *bij-betw* (*inverse* $x \cdot y$) $_R\ (H \mid \cdot\ x)\ (H \mid \cdot\ y)$
 $\langle \text{proof} \rangle$

p 52, ll 25–26

theorem *Right-Cosets-cardinality*:
 $\llbracket x \in G; y \in G \rrbracket \implies \text{card } (H \mid \cdot\ x) = \text{card } (H \mid \cdot\ y)$
 $\langle \text{proof} \rangle$

p 52, l 27

theorem *Right-Coset-unit*:
 $H \mid \cdot\ \mathbf{1} = H$

$\langle proof \rangle$

p 52, l 27

theorem *Right-Coset-cardinality:*

$x \in G \implies \text{card } (H \mid \cdot x) = \text{card } H$

$\langle proof \rangle$

p 52, ll 31–32

definition *index = card orbit.Partition*

Theorem 1.5

p 52, ll 33–35; p 53, ll 1–2

theorem *lagrange:*

assumes *finite G*

shows $\text{card } G = \text{card } H * \text{index}$

$\langle proof \rangle$

end

lemma *subgroup-of-groupI:*

assumes *Subgroup H G composition unit and Group G composition unit*

shows *subgroup-of-group H G composition unit*

$\langle proof \rangle$

Left cosets

context *Subgroup* **begin**

p 31, ll 11–12

lemma *image-of-inverse [intro, simp]:*

$x \in H \implies x \in \text{inverse } ' H$

$\langle proof \rangle$

lemma *image-of-inverse-iff:*

$x \in \text{inverse } ' H \longleftrightarrow x \in H$

$\langle proof \rangle$

end

context *Group* **begin**

p 53, ll 6–7

lemma *inverse-subgroupI:*

assumes *sub: Subgroup H G ($\cdot$) 1*

shows *Subgroup (inverse ' H) G ($\cdot$) 1*

$\langle proof \rangle$

p 53, ll 6–7

lemma *inverse-subgroupD*:
assumes *sub*: *Subgroup* (*inverse* ‘ *H*) *G* ($\cdot$) **1**
and *inv*: $H \subseteq \text{Units}$
shows *Subgroup* *H* *G* ($\cdot$) **1**
 $\langle \text{proof} \rangle$

Intersections are the basic way of restricting subgroups and normal subgroups to a smaller ambient group.

lemma *subgroup-intersection*:
assumes *Subgroup* *H* *G* ($\cdot$) **1** *Subgroup* *K* *G* ($\cdot$) **1**
shows *Subgroup* ($H \cap K$) *G* ($\cdot$) **1**
 $\langle \text{proof} \rangle$

lemma *commute-iff-inverse*:
assumes $g \in G$ $h \in G$
shows $(g \cdot h = h \cdot g) \longleftrightarrow (g \cdot h \cdot \text{local.inverse } g = h)$
 $\langle \text{proof} \rangle$

end

context *subgroup-of-group* **begin**

p 53, l 6

interpretation *right-translations-of-group* $\langle \text{proof} \rangle$

translation ‘ *H* denotes Jacobson’s $H_R(G)$ and *Translations* denotes Jacobson’s G_R .

p 53, ll 6–7

theorem *right-translations-of-subgroup-are-transformation-group* [intro]:
transformation-group (*translation* ‘ *H*) *G*
 $\langle \text{proof} \rangle$

p 53, ll 6–7

interpretation *transformation-group translation* ‘ *H* *G* $\langle \text{proof} \rangle$

Equation 23 for left cosets

p 53, ll 7–8

theorem *Left-Coset-is-orbit*:
 $x \in G \implies x \cdot | H = \text{orbit.Class } x$
 $\langle \text{proof} \rangle$

end

1.7 Congruences. Quotient Monoids and Groups

Def 1.4

p 54, ll 19–22

locale *Monoid-congruence* = *Monoid* + *Equivalence* **where** $S = M +$
assumes *cong*: $\llbracket (a, a') \in E; (b, b') \in E \rrbracket \implies (a \cdot b, a' \cdot b') \in E$
begin

p 54, ll 26–28

theorem *Class-cong*:

$\llbracket \text{Class } a = \text{Class } a'; \text{Class } b = \text{Class } b'; a \in M; a' \in M; b \in M; b' \in M \rrbracket \implies$
 $\text{Class } (a \cdot b) = \text{Class } (a' \cdot b')$
<proof>

p 54, ll 28–30

definition *quotient-composition* (**infixl** $\langle \cdot \rangle$ 70)

where *quotient-composition* = $(\lambda A \in M / E. \lambda B \in M / E. \text{THE } C. \exists a \in A. \exists b \in B. C = \text{Class } (a \cdot b))$

p 54, ll 28–30

theorem *Class-commutes-with-composition*:

$\llbracket a \in M; b \in M \rrbracket \implies \text{Class } a [\cdot] \text{Class } b = \text{Class } (a \cdot b)$
<proof>

p 54, ll 30–31

theorem *quotient-composition-closed* [*intro*, *simp*]:

$\llbracket A \in M / E; B \in M / E \rrbracket \implies A [\cdot] B \in M / E$
<proof>

p 54, l 32; p 55, ll 1–3

sublocale *quotient*: *Monoid* M / E ($[\cdot]$) *Class* **1**
<proof>

end

p 55, ll 16–17

locale *group-congruence* = *Group* + *Monoid-congruence* **where** $M = G$ **begin**

p 55, ll 16–17

notation *quotient-composition* (**infixl** $\langle \cdot \rangle$ 70)

p 55, l 18

theorem *Class-right-inverse*:

$a \in G \implies \text{Class } a [\cdot] \text{Class } (\text{inverse } a) = \text{Class } \mathbf{1}$
<proof>

p 55, l 18

theorem *Class-left-inverse*:

$a \in G \implies \text{Class } (\text{inverse } a) [\cdot] \text{Class } a = \text{Class } \mathbf{1}$

$\langle proof \rangle$

p 55, l 18

theorem *Class-invertible:*

$a \in G \implies \text{quotient.invertible } (\text{Class } a)$

$\langle proof \rangle$

p 55, l 18

theorem *Class-commutes-with-inverse:*

$a \in G \implies \text{quotient.inverse } (\text{Class } a) = \text{Class } (\text{inverse } a)$

$\langle proof \rangle$

p 55, l 17

sublocale *quotient: Group G / E ($[\cdot]$) Class $\mathbf{1}$*

$\langle proof \rangle$

end

Def 1.5

p 55, ll 22–25

locale *normal-subgroup =*

*subgroup-of-group $K \ G \ (\cdot) \ \mathbf{1}$ for K and G and composition (**infixl** $\langle \cdot \rangle$ 70) and unit*

($\langle \mathbf{1} \rangle$) +

assumes *normal: $\llbracket g \in G; k \in K \rrbracket \implies \text{inverse } g \cdot k \cdot g \in K$*

context *Group begin*

The trivial subgroup is normal in every group.

lemma *trivial-normal-subgroup:*

normal-subgroup $\{\mathbf{1}\} \ G \ (\cdot) \ \mathbf{1}$

$\langle proof \rangle$

end

Lemmas from the proof of Thm 1.6

context *subgroup-of-group begin*

We use H for K .

p 56, ll 14–16

theorem *Left-equals-Right-coset-implies-normality:*

assumes *[simp]: $\bigwedge g. g \in G \implies g \cdot | H = H \cdot | g$*

shows *normal-subgroup $H \ G \ (\cdot) \ \mathbf{1}$*

$\langle proof \rangle$

end

Thm 1.6, first part

context *group-congruence* **begin**

Jacobson's K

p 56, l 29

definition $Normal = Class\ 1$

p 56, ll 3–6

interpretation $Subgroup\ Normal\ G\ (\cdot)\ 1$
 $\langle proof \rangle$

Coset notation

p 56, ll 5–6

interpretation $subgroup-of-group\ Normal\ G\ (\cdot)\ 1$ $\langle proof \rangle$

Equation 25 for right cosets

p 55, ll 29–30; p 56, ll 6–11

theorem *Right-Coset-Class-unit:*

assumes $g: g \in G$ **shows** $Normal\ |\cdot\ g = Class\ g$
 $\langle proof \rangle$

Equation 25 for left cosets

p 55, ll 29–30; p 56, ll 6–11

theorem *Left-Coset-Class-unit:*

assumes $g: g \in G$ **shows** $g\cdot| Normal = Class\ g$
 $\langle proof \rangle$

Thm 1.6, statement of first part

p 55, ll 28–29; p 56, ll 12–16

theorem *Class-unit-is-normal:*

$normal-subgroup\ Normal\ G\ (\cdot)\ 1$
 $\langle proof \rangle$

sublocale $normal: normal-subgroup\ Normal\ G\ (\cdot)\ 1$
 $\langle proof \rangle$

end

context *normal-subgroup* **begin**

p 56, ll 16–19

theorem *Left-equals-Right-coset:*

$g \in G \implies g\cdot| K = K\ |\cdot\ g$

<proof>

Thm 1.6, second part

p 55, ll 31–32; p 56, ll 20–21

definition *Congruence* = $\{(a, b). a \in G \wedge b \in G \wedge \text{inverse } a \cdot b \in K\}$

p 56, ll 21–22

interpretation *right-translations-of-group* *<proof>*

p 56, ll 21–22

interpretation *transformation-group translation* ‘*K G rewrites Orbit-Relation = Congruence*

<proof>

p 56, ll 20–21

lemma *CongruenceI*: $\llbracket a = b \cdot k; a \in G; b \in G; k \in K \rrbracket \implies (a, b) \in \text{Congruence}$

<proof>

p 56, ll 20–21

lemma *CongruenceD*: $(a, b) \in \text{Congruence} \implies \exists k \in K. a = b \cdot k$

<proof>

“We showed in the last section that the relation we are considering is an equivalence relation in G for any subgroup K of G . We now proceed to show that normality of K ensures that $[...] a \equiv b \pmod{K}$ is a congruence.”

p 55, ll 30–32; p 56, ll 1, 22–28

sublocale *group-congruence* **where** $E = \text{Congruence}$ **rewrites** $\text{Normal} = K$

<proof>

lemma *set-prod-commute*:

assumes $S \subseteq G$

shows $(\text{case-prod}(\cdot)) \text{ ‘ } (S \times K) = (\text{case-prod}(\cdot)) \text{ ‘ } (K \times S)$

<proof>

p 56, ll 31–32; no infix syntax because it leads to many problems

abbreviation *Factor-Group*

where $\text{Factor-Group} \equiv G / (\text{normal-subgroup.Congruence } K \ G \ (\cdot) \ 1)$

end

context *Group* **begin**

lemma *normal-subgroup-intersection*:

assumes K : *normal-subgroup* $K \ G \ (\cdot) \ 1$

and H : *Subgroup* $H \ G \ (\cdot) \ 1$

shows *normal-subgroup* $(H \cap K) \ H \ (\cdot) \ 1$

$\langle proof \rangle$

end

context *normal-subgroup* **begin**

p 56, ll 28–29

theorem *Class-unit-normal-subgroup*: $Class\ 1 = K$
 $\langle proof \rangle$

The quotient by a normal subgroup is trivial exactly when that subgroup is the whole carrier. Keeping the representative argument here avoids repeating it for each client that packages a quotient group.

lemma *Factor-Group-eq-singleton-iff*:
 $Factor-Group = \{Class\ 1\} \longleftrightarrow K = G$
 $\langle proof \rangle$

p 56, ll 1–2; p 56, l 29

theorem *Class-is-Left-Coset*:
 $g \in G \implies Class\ g = g \cdot | K$
 $\langle proof \rangle$

p 56, l 29

lemma *Left-CosetE*: $\llbracket A \in Factor-Group; \bigwedge a. a \in G \implies P\ (a \cdot | K) \rrbracket \implies P\ A$
 $\langle proof \rangle$

Equation 26

p 56, ll 32–34

theorem *factor-composition* $[simp]$:
 $\llbracket g \in G; h \in G \rrbracket \implies (g \cdot | K) [\cdot] (h \cdot | K) = g \cdot h \cdot | K$
 $\langle proof \rangle$

p 56, l 35

theorem *factor-unit*:
 $K = 1 \cdot | K$
 $\langle proof \rangle$

p 56, l 35

theorem *factor-inverse* $[simp]$:
 $g \in G \implies quotient.inverse\ (g \cdot | K) = (inverse\ g \cdot | K)$
 $\langle proof \rangle$

lemma *Factor-Group-eq-Left-Cosets*:
 $Factor-Group = (\lambda g. g \cdot | K) \cdot G$
 $\langle proof \rangle$

lemma *Factor-Group-eq-Right-Cosets:*

Factor-Group = $(\lambda g. K \mid \cdot g)$ ‘ *G*
 ⟨proof⟩

lemma *Class-cardinality:*

assumes $g \in G$
shows $\text{card } (\text{Class } g) = \text{card } K$
 ⟨proof⟩

lemma *card-Factor-Group:*

assumes *finite G*
shows $\text{card } (\text{Factor-Group}) = \text{index}$
 ⟨proof⟩

lemma *quotient-composition-PiE:*

quotient-composition $\in \text{Factor-Group} \rightarrow_E \text{Factor-Group} \rightarrow_E \text{Factor-Group}$
 ⟨proof⟩

end

p 57, ll 4–5

locale *subgroup-of-abelian-group* = *subgroup-of-group* $H \ G \ (\cdot) \ \mathbf{1} + \text{Abelian-Group } G$
 $(\cdot) \ \mathbf{1}$

for H **and** G **and** *composition* (infixl $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$)

p 57, ll 4–5

sublocale *subgroup-of-abelian-group* $\subseteq$ *normal-subgroup* $H \ G \ (\cdot) \ \mathbf{1}$
 ⟨proof⟩

context *subgroup-of-abelian-group* **begin**

lemma *quotient-abelian:* *Abelian-Group* *Factor-Group* *quotient-composition* (*Class* $\mathbf{1}$)

⟨proof⟩

end

1.8 Homomorphisms

Def 1.6

p 58, l 33; p 59, ll 1–2

locale *Monoid-homomorphism* =

map $\eta \ M \ M' + \text{source: Monoid } M \ (\cdot) \ \mathbf{1} + \text{target: Monoid } M' \ (\cdot') \ \mathbf{1}'$

for η **and** M **and** *composition* (infixl $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$)

and M' **and** *composition'* (infixl $\langle \cdot' \rangle$ 70) **and** *unit'* ($\langle \mathbf{1}' \rangle$) +

assumes *commutes-with-composition:* $\llbracket x \in M; y \in M \rrbracket \implies \eta (x \cdot y) = \eta x \cdot' \eta y$

and *commutes-with-unit:* $\eta \ \mathbf{1} = \mathbf{1}'$

begin

Jacobson notes that *commutes-with-unit* is not necessary for groups, but doesn't make use of that later.

p 58, l 33; p 59, ll 1–2

notation *source.invertible* ($\langle \text{invertible} \rightarrow [100] \ 100 \rangle$)

notation *source.inverse* ($\langle \text{inverse} \rightarrow [100] \ 100 \rangle$)

notation *target.invertible* ($\langle \text{invertible}'' \rightarrow [100] \ 100 \rangle$)

notation *target.inverse* ($\langle \text{inverse}'' \rightarrow [100] \ 100 \rangle$)

end

p 59, ll 29–30

locale *Monoid-epimorphism* = *Monoid-homomorphism* + *surjective-map* $\eta \ M \ M'$

p 59, l 30

locale *Monoid-monomorphism* = *Monoid-homomorphism* + *injective-map* $\eta \ M \ M'$

p 59, ll 30–31

sublocale *Monoid-isomorphism* $\subseteq$ *Monoid-epimorphism*

$\langle \text{proof} \rangle$

p 59, ll 30–31

sublocale *Monoid-isomorphism* $\subseteq$ *Monoid-monomorphism*

$\langle \text{proof} \rangle$

context *Monoid-homomorphism* **begin**

p 59, ll 33–34

theorem *invertible-image-lemma*:

assumes *invertible* $a \ a \in M$

shows $\eta \ a \cdot' \eta \ (\text{inverse } a) = \mathbf{1}'$ and $\eta \ (\text{inverse } a) \cdot' \eta \ a = \mathbf{1}'$

$\langle \text{proof} \rangle$

p 59, l 34; p 60, l 1

theorem *invertible-target-invertible* [*intro*, *simp*]:

$\llbracket \text{invertible } a; a \in M \rrbracket \implies \text{invertible}' (\eta \ a)$

$\langle \text{proof} \rangle$

p 60, l 1

theorem *invertible-commutes-with-inverse*:

$\llbracket \text{invertible } a; a \in M \rrbracket \implies \eta \ (\text{inverse } a) = \text{inverse}' (\eta \ a)$

$\langle \text{proof} \rangle$

end

p 60, ll 32–34; p 61, l 1

sublocale *Monoid-congruence*

$\subseteq$ *natural: Monoid-homomorphism Class $M (\cdot) \mathbf{1} M / E ([\cdot])$ Class $\mathbf{1}$*
 $\langle proof \rangle$

Fundamental Theorem of Homomorphisms of Monoids

p 61, ll 5, 14–16

sublocale *Monoid-homomorphism $\subseteq$ image: Submonoid $\eta \text{ ‘ } M M' (\cdot) \mathbf{1}'$*
 $\langle proof \rangle$

p 61, l 4

locale *Monoid-homomorphism-fundamental = Monoid-homomorphism*
begin

p 61, ll 17–18

sublocale *fiber-relation $\eta M M' \langle proof \rangle$*
notation *Fiber-Relation $(\lrcorner E'(-) \rhd)$*

p 61, ll 6–7, 18–20

sublocale *Monoid-congruence* **where** $E = E(\eta)$
 $\langle proof \rangle$

p 61, ll 7–9

induced denotes Jacobson’s $\bar{\eta}$. We have the commutativity of the diagram,
 where *induced* is unique:

compose M induced Class = η

$\llbracket ?\beta \in \text{Partition} \rightarrow_E M'; \text{compose } M ?\beta \text{ Class} = \eta \rrbracket \implies ?\beta = \text{induced}$

.

p 61, l 20

notation *quotient-composition (infixl $\lrcorner [\cdot] \rhd$ 70)*

p 61, ll 7–8, 22–25

sublocale *induced: Monoid-homomorphism induced $M / E(\eta) ([\cdot])$ Class $\mathbf{1} M' (\cdot) \mathbf{1}'$*
 $\langle proof \rangle$

p 61, ll 9, 26

sublocale *natural: Monoid-epimorphism Class $M (\cdot) \mathbf{1} M / E(\eta) ([\cdot])$ Class $\mathbf{1}$* $\langle proof \rangle$

p 61, ll 9, 26–27

sublocale *induced: Monoid-monomorphism induced $M / E(\eta) ([\cdot])$ Class $\mathbf{1} M' (\cdot) \mathbf{1}'$*
 $\langle proof \rangle$

end

p 62, ll 12–13

locale *group-homomorphism* =
Monoid-homomorphism $\eta \ G \ (\cdot) \ \mathbf{1} \ G' \ (\cdot) \ \mathbf{1}' +$
source: Group $G \ (\cdot) \ \mathbf{1} +$ *target: Group* $G' \ (\cdot) \ \mathbf{1}'$
for η **and** G **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$)
and G' **and** *composition'* (**infixl** $\langle \cdot' \rangle$ 70) **and** *unit'* ($\langle \mathbf{1}' \rangle$)
begin

p 62, l 13

sublocale *image: Subgroup* $\eta \text{ ' } G \ G' \ (\cdot) \ \mathbf{1}'$
 $\langle \text{proof} \rangle$

p 62, ll 13–14

definition $Ker = \eta \text{ ' } \{\mathbf{1}'\} \cap G$

p 62, ll 13–14

lemma *Ker-equality:*
 $Ker = \{a \mid a. a \in G \wedge \eta \ a = \mathbf{1}'\}$
 $\langle \text{proof} \rangle$

p 62, ll 13–14

lemma *Ker-closed* [*intro, simp*]:
 $a \in Ker \implies a \in G$
 $\langle \text{proof} \rangle$

p 62, ll 13–14

lemma *Ker-image* [*intro*]:
 $a \in Ker \implies \eta \ a = \mathbf{1}'$
 $\langle \text{proof} \rangle$

p 62, ll 13–14

lemma *Ker-memI* [*intro*]:
 $\llbracket \eta \ a = \mathbf{1}'; a \in G \rrbracket \implies a \in Ker$
 $\langle \text{proof} \rangle$

p 62, ll 15–16

sublocale *kernel: normal-subgroup* $Ker \ G$
 $\langle \text{proof} \rangle$

p 62, ll 17–20

theorem *injective-iff-kernel-unit:*
 $inj\text{-}on \ \eta \ G \longleftrightarrow Ker = \{\mathbf{1}\}$
 $\langle \text{proof} \rangle$

end

p 62, l 24

locale *group-epimorphism* = *group-homomorphism* + *Monoid-epimorphism* η *G* ($\cdot$)
1 *G'* ($\cdot$) **1'**

p 62, l 21

locale *normal-subgroup-in-kernel* =
group-homomorphism + *contained: normal-subgroup* *L G* ($\cdot$) **1** **for** *L* +
assumes subset: L $\subseteq$ *Ker*
begin

p 62, l 21

notation *contained.quotient-composition* (**infixl** $\langle[\cdot]\rangle$ 70)

"homomorphism onto G/L "

p 62, ll 23–24

sublocale *natural: group-epimorphism contained.Class G* ($\cdot$) **1** *contained.Factor-Group*
 $([\cdot])$ *contained.Class 1* $\langle$ *proof* $\rangle$

p 62, ll 25–26

theorem *left-coset-equality*:
assumes eq: a $\cdot$ *L* = *b* $\cdot$ *L* **and** [*simp*]: *a* $\in$ *G* **and** *b*: *b* $\in$ *G*
shows η *a* = η *b*
 $\langle$ *proof* $\rangle$

$\bar{\eta}$

p 62, ll 26–27

definition *induced* = ($\lambda A \in$ *contained.Factor-Group*. *THE* *b*. $\exists a \in G$. *a* $\cdot$ *L* = *A* $\wedge$
b = η *a*)

p 62, ll 26–27

lemma *induced-closed* [*intro*, *simp*]:
assumes [*simp*]: *A* $\in$ *contained.Factor-Group* **shows** *induced A* $\in$ *G'*
 $\langle$ *proof* $\rangle$

p 62, ll 26–27

lemma *induced-undefined* [*intro*, *simp*]:
A $\notin$ *contained.Factor-Group* $\implies$ *induced A* = *undefined*
 $\langle$ *proof* $\rangle$

p 62, ll 26–27

theorem *induced-left-coset-closed* [*intro*, *simp*]:
a $\in$ *G* $\implies$ *induced (a* $\cdot$ *L)* $\in$ *G'*
 $\langle$ *proof* $\rangle$

p 62, ll 26–27

theorem *induced-left-coset-equality* [*simp*]:

assumes [simp]: $a \in G$ **shows** $\text{induced } (a \cdot | L) = \eta \ a$
 $\langle \text{proof} \rangle$

p 62, l 27

theorem *induced-Left-Coset-commutes-with-composition* [simp]:
 $\llbracket a \in G; b \in G \rrbracket \implies \text{induced } ((a \cdot | L) [|] (b \cdot | L)) = \text{induced } (a \cdot | L) \cdot' \text{induced } (b \cdot | L)$
 $\langle \text{proof} \rangle$

p 62, ll 27–28

theorem *induced-group-homomorphism*:
 $\text{group-homomorphism induced contained.Factor-Group } ([\cdot]) \text{ (contained.Class } \mathbf{1}) \ G' \ (\cdot') \ \mathbf{1}'$
 $\langle \text{proof} \rangle$

p 62, l 28

sublocale *induced*: $\text{group-homomorphism induced contained.Factor-Group } ([\cdot]) \text{ contained.Class } \mathbf{1} \ G' \ (\cdot') \ \mathbf{1}'$
 $\langle \text{proof} \rangle$

p 62, ll 28–29

theorem *factorization-lemma*: $a \in G \implies \text{compose } G \text{ induced contained.Class } a = \eta \ a$
 $\langle \text{proof} \rangle$

p 62, ll 29–30

theorem *factorization* [simp]: $\text{compose } G \text{ induced contained.Class} = \eta$
 $\langle \text{proof} \rangle$

Jacobson does not state the uniqueness of *induced* explicitly but he uses it later, for rings, on p 107.

p 62, l 30

theorem *uniqueness*:
assumes $\text{map: } \beta \in \text{contained.Factor-Group} \rightarrow_E G'$
and $\text{factorization: compose } G \ \beta \text{ contained.Class} = \eta$
shows $\beta = \text{induced}$
 $\langle \text{proof} \rangle$

p 62, l 31

theorem *induced-image*:
 $\text{induced } \cdot \text{ contained.Factor-Group} = \eta \cdot G$
 $\langle \text{proof} \rangle$

p 62, l 33

interpretation L : *normal-subgroup* $L \text{ Ker}$
 $\langle \text{proof} \rangle$

p 62, ll 31–33

theorem *induced-kernel*:

induced.Ker = Ker / L.Congruence
<proof>

p 62, ll 34–35

theorem *induced-inj-on*:

inj-on induced contained.Factor-Group $\longleftrightarrow L = Ker$
<proof>

end

Fundamental Theorem of Homomorphisms of Groups

p 63, l 1

locale *group-homomorphism-fundamental = group-homomorphism* **begin**

p 63, l 1

notation *kernel.quotient-composition* (**infixl** $\langle[\cdot]\rangle$ 70)

p 63, l 1

sublocale *normal-subgroup-in-kernel* **where** $L = Ker$ *<proof>*

p 62, ll 36–37; p 63, l 1

induced denotes Jacobson’s $\bar{\eta}$. We have the commutativity of the diagram,
where *induced* is unique:

compose G induced kernel.Class = η

$\llbracket ?\beta \in kernel.Factor-Group \rightarrow_E G'; compose\ G\ ?\beta\ kernel.Class = \eta \rrbracket$
 $\implies ?\beta = induced$

end

p 63, l 5

locale *group-isomorphism = group-homomorphism + bijective-map $\eta\ G\ G'$* **begin**

p 63, l 5

sublocale *Monoid-isomorphism $\eta\ G\ (\cdot)\ 1\ G'\ (\cdot)\ 1'$*
<proof>

p 63, l 6

lemma *inverse-group-isomorphism*:

group-isomorphism (restrict (inv-into $G\ \eta$) G') $G'\ (\cdot)\ 1'\ G\ (\cdot)\ 1$
<proof>

end

p 63, l 6

definition *isomorphic-as-groups* (**infixl** $\langle \cong_G \rangle$ 50)

where $\mathcal{G} \cong_G \mathcal{G}' \longleftrightarrow (\text{let } (G, \text{composition}, \text{unit}) = \mathcal{G}; (G', \text{composition}', \text{unit}') = \mathcal{G}' \text{ in}$
 $(\exists \eta. \text{group-isomorphism } \eta \ G \ \text{composition} \ \text{unit} \ G' \ \text{composition}' \ \text{unit}'))$

p 63, l 6

lemma *isomorphic-as-groups-symmetric:*

$(G, \text{composition}, \text{unit}) \cong_G (G', \text{composition}', \text{unit}') \implies (G', \text{composition}', \text{unit}') \cong_G (G, \text{composition}, \text{unit})$
 $\langle \text{proof} \rangle$

lemma *isomorphic-as-groups-transitive:*

assumes $(G, \text{composition}, \text{unit}) \cong_G (G', \text{composition}', \text{unit}')$
and $(G', \text{composition}', \text{unit}') \cong_G (G'', \text{composition}'', \text{unit}'')$
shows $(G, \text{composition}, \text{unit}) \cong_G (G'', \text{composition}'', \text{unit}'')$
 $\langle \text{proof} \rangle$

On groups the two notions of isomorphism coincide: a bijective monoid isomorphism between groups is already a group isomorphism, since *group-homomorphism* adds nothing to *Monoid-homomorphism* beyond the two structures being groups. So isomorphism classes may be formed once at the level of monoids without losing the group-level statement.

lemma *isomorphic-as-monoids-iff-groups:*

assumes G : Group G composition unit **and** G' : Group G' composition' unit'
shows $(G, \text{composition}, \text{unit}) \cong_M (G', \text{composition}', \text{unit}')$
 $\longleftrightarrow (G, \text{composition}, \text{unit}) \cong_G (G', \text{composition}', \text{unit}')$
 $\langle \text{proof} \rangle$

p 63, l 1

sublocale *group-isomorphism* $\subseteq$ *group-epimorphism* $\langle \text{proof} \rangle$

p 63, l 1

locale *group-epimorphism-fundamental* = *group-homomorphism-fundamental* + *group-epimorphism*
begin

p 63, ll 1–2

interpretation *image: group-homomorphism induced kernel.Factor-Group* $([.])$ *kernel.Class* $\mathbf{1} \ (\eta \cdot G) \ (\cdot) \ \mathbf{1}'$
 $\langle \text{proof} \rangle$

p 63, ll 1–2

sublocale *image: group-isomorphism induced kernel.Factor-Group* $([.])$ *kernel.Class* $\mathbf{1} \ (\eta \cdot G) \ (\cdot) \ \mathbf{1}'$
 $\langle \text{proof} \rangle$

end

context *group-homomorphism* **begin**

p 63, ll 5–7

theorem *image-isomorphic-to-factor-group*:

$\exists K$ *composition* *unit*. *normal-subgroup* $K \ G \ (\cdot) \ \mathbf{1} \wedge (\eta \text{ ‘ } G, (\cdot), \mathbf{1}) \cong_G (G /$
(normal-subgroup.Congruence $K \ G \ (\cdot) \ \mathbf{1})$, *composition*, *unit*)
<proof>

p 65

The First Isomorphism Theorem states that for any group homomorphism $\eta : G \rightarrow G'$, the quotient $G/\ker(\eta)$ is isomorphic to the image $im(\eta)$. The isomorphism is given by the induced map $gK \mapsto \eta(g)$, which is well-defined because elements in the same coset of $\ker(\eta)$ have the same image under η .

theorem *first-isomorphism*:

(kernel.Factor-Group, *kernel.quotient-composition*, *kernel.Class* $\mathbf{1}) \cong_G (\eta \text{ ‘ } G, (\cdot),$
 $\mathbf{1})$
<proof>

end

1.9 Correspondence Theorem and Third Isomorphism Theorem

p 65

The proofs in this section, in their initial form, were generated by Claude Opus 4.6 through Dominic Mulligan’s Isabelle Assistant. Then simplified and tidied by LCP.

Let K be a normal subgroup of G and H a subgroup of G containing K . Then H/K is a subgroup of G/K , the map $H \mapsto H/K$ is a bijection between subgroups of G containing K and subgroups of G/K , H is normal in G if and only if H/K is normal in G/K , and $G/H \cong (G/K)/(H/K)$.

locale *normal-subgroup-in-subgroup* =

K: *normal-subgroup* $K \ G \ (\cdot) \ \mathbf{1} +$

H: *subgroup-of-group* $H \ G \ (\cdot) \ \mathbf{1}$

for *K* **and** *H* **and** *G* **and** *composition* (*infixl* $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$) +

assumes *K-contained*: $K \subseteq H$

begin

interpretation *K-subgroup-H*: *Subgroup* $K \ H \ (\cdot) \ \mathbf{1}$

<proof>

K is a normal subgroup of H .

interpretation *K-in-H*: normal-subgroup $K\ H\ (\cdot)\ \mathbf{1}$
 $\langle proof \rangle$

notation *K.quotient-composition* (**infixl** $\langle [\cdot] \rangle$ 70)

H/K as a subset of G/K , defined as the image of H under the canonical projection.

definition *Fract-H-K* = $K.Class\ 'H$

One might expect the quotient H/K to be $K.Factor-Group$ with H in place of G . However, $K.Factor-Group$ always forms the congruence of K relative to G (not H), so substituting H for the carrier would give $H / K.Congruence$. Since $K.Congruence$ is an equivalence on G rather than H , the conditional defining equation of $(/)$ cannot be applied, leaving the term opaque.

The correct formalisation of H/K is $K-in-H.Factor-Group$, the partition of H under $K-in-H.Congruence$ —the congruence of K viewed as a normal subgroup of H . The proof that *Fract-H-K* equals $K-in-H.Factor-Group$ follows from the fact that $K.Class$ and $K-in-H.Class$ agree on elements of H : both map $h \in H$ to the left coset hK , by $K.Class-is-Left-Coset$ and $K-in-H.Class-is-Left-Coset$.

lemma *H-Class-eq*: $h \in H \implies K-in-H.Class\ h = K.Class\ h$
 $\langle proof \rangle$

lemma *Fract-H-K-is-quotient*: $Fract-H-K = K-in-H.Partition$
 $\langle proof \rangle$

lemma *Fract-H-K-subset*: $Fract-H-K \subseteq K.Factor-Group$
 $\langle proof \rangle$

lemma *Class-H-closed* [*intro*, *simp*]:
 $h \in H \implies K.Class\ h \in Fract-H-K$
 $\langle proof \rangle$

lemma *Fract-H-K-unit-closed* [*intro*, *simp*]:
 $K.Class\ \mathbf{1} \in Fract-H-K$
 $\langle proof \rangle$

lemma *Fract-H-K-memE* [*elim*]:
 $\llbracket A \in Fract-H-K; \bigwedge h. \llbracket h \in H; A = K.Class\ h \rrbracket \implies P \rrbracket \implies P$
 $\langle proof \rangle$

H/K is a subgroup of G/K .

lemma *Fract-H-K-into-subset-H*: $h \in Fract-H-K \implies h \subseteq H$
 $\langle proof \rangle$

theorem *Fract-H-K-subgroup*: Subgroup $Fract-H-K\ K.Factor-Group\ ([\cdot])\ (K.Class\ \mathbf{1})$
 $\langle proof \rangle$

sublocale *Fract-H-K*: subgroup-of-group $Fract-H-K\ K.Factor-Group\ ([\cdot])\ K.Class\ \mathbf{1}$

$\langle proof \rangle$

H is normal in G if and only if H/K is normal in G/K .

theorem *normal-iff*:

normal-subgroup $H\ G\ (\cdot)\ \mathbf{1} \longleftrightarrow \text{normal-subgroup } Fract\text{-}H\text{-}K\ K.\text{Factor-Group } ([\cdot])$
(K.Class 1)
 $\langle proof \rangle$

Third Isomorphism Theorem: $G/H \cong (G/K)/(H/K)$

We assume H is normal in G and construct a surjective group homomorphism from G/K to G/H whose kernel is H/K .

theorem *third-isomorphism*:

assumes *H-normal*: *normal-subgroup* $H\ G\ (\cdot)\ \mathbf{1}$
defines $CongH \equiv \text{normal-subgroup.Congruence } H\ G\ (\cdot)\ \mathbf{1}$
and $CongHK \equiv \text{normal-subgroup.Congruence } Fract\text{-}H\text{-}K\ K.\text{Factor-Group } K.\text{quotient-composition}$
(K.Class 1)
shows $(G / CongH,$
 $Monoid\text{-congruence.quotient-composition } G\ (\cdot)\ CongH,$
 $Equivalence.Class\ G\ CongH\ \mathbf{1}) \cong_G$
 $(K.\text{Factor-Group} / CongHK,$
 $Monoid\text{-congruence.quotient-composition } K.\text{Factor-Group } K.\text{quotient-composition}$
 $CongHK,$
 $Equivalence.Class\ K.\text{Factor-Group } CongHK\ (K.Class\ \mathbf{1}))$
 $\langle proof \rangle$

end

The product of a subgroup and a normal subgroup is again a subgroup. We keep the construction set-based, in keeping with the rest of this development, and package its closure facts in a locale so that later isomorphism theorems can use the product as an ambient group in its own right.

locale *normal-subgroup-product* =

K : *normal-subgroup* $K\ G\ (\cdot)\ \mathbf{1}$ +

H : *subgroup-of-group* $H\ G\ (\cdot)\ \mathbf{1}$

for K **and** H **and** G **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$)

begin

definition HK **where** $HK \equiv (case\text{-}prod(\cdot))\ ' (H \times K)$

lemma *HK-memI*:

$\llbracket h \in H; k \in K \rrbracket \implies h \cdot k \in HK$

$\langle proof \rangle$

lemma *HK-memE*:

assumes $x \in HK$

obtains $h\ k$ **where** $h \in H\ k \in K\ x = h \cdot k$

$\langle proof \rangle$

lemma *HK-eq-KH*: $HK = (case-prod(\cdot)) \text{ ' } (K \times H)$
 ⟨proof⟩

interpretation *HK-mon*: *Monoid* HK
 ⟨proof⟩

interpretation *HK-sub*: *Subgroup* $HK \ G$
 ⟨proof⟩

lemma *HK-subgroup*: *Subgroup* $HK \ G \ (\cdot) \ 1$
 ⟨proof⟩

lemma *H-subset-HK*: $H \subseteq HK$
 ⟨proof⟩

lemma *K-subset-HK*: $K \subseteq HK$
 ⟨proof⟩

end

locale *second-iso-theorem* = *normal-subgroup-product* $K \ H \ G \ (\cdot) \ 1$
for K **and** H **and** G **and** *composition* (*infixl* $\langle \cdot \rangle$ 70) **and** *unit* ($\langle 1 \rangle$)
begin

interpretation *HIntK-Mon*: *Monoid* $H \cap K$
 ⟨proof⟩

interpretation *HIntKn*: *normal-subgroup* $H \cap K \ H$
 ⟨proof⟩

Defining the quotient groups as sets of equivalence classes

definition *Fract-HK-K* $\equiv K.Class \text{ ' } HK$

definition *Fract-H-HIntK* $\equiv HIntKn.Class \text{ ' } H$

definition φ **where** $\varphi \equiv (\lambda A \in Fract-HK-K. \text{ THE } B. \exists g \in HK. A = K.Class \ g \wedge B = HIntKn.Class \ g)$

theorem *second-isomorphism*:

shows $(Fract-HK-K, K.quotient-composition, K.Class \ 1) \cong_G$
 $(Fract-H-HIntK, HIntKn.quotient-composition, HIntKn.Class \ 1)$

⟨proof⟩

end

1.10 Dropping from elements of a partition to representatives

This prevents a change of type for inductive proofs that involve quotienting.

locale *class-representatives* =
fixes $A :: 'a \text{ set}$ **and** P **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle 1 \rangle$)
assumes P : *partition-on* A P
assumes *comp*: $(\cdot) \in P \rightarrow_E P \rightarrow_E P$
assumes *unit*: $1 \in P$
begin

definition *the-part* :: $'a \Rightarrow 'a \text{ set}$ **where**
the-part $x \equiv \text{THE } X. X \in P \wedge x \in X$

definition *rep-comp* :: $'a \Rightarrow 'a \Rightarrow 'a$ **where**
rep-comp $\equiv \lambda x \in A. \lambda y \in A. \text{some-elem } ((\text{the-part } x) \cdot (\text{the-part } y))$

lemma *some-elem-typing*: $X \in P \implies \text{some-elem } X \in A$
 $\langle \text{proof} \rangle$

lemma *the-part-works*:
assumes $x \in A$ **shows** $x \in \text{the-part } x$ *the-part* $x \in P$
 $\langle \text{proof} \rangle$

lemma *typing*: $\llbracket x \in A; y \in A \rrbracket \implies \text{some-elem } (\text{the-part } x \cdot \text{the-part } y) \in A$
 $\langle \text{proof} \rangle$

lemma *unit-typing*: $\text{some-elem } 1 \in A$
 $\langle \text{proof} \rangle$

lemma *rep-comp-typing*: $\text{rep-comp} \in A \rightarrow_E (A \rightarrow_E A)$
 $\langle \text{proof} \rangle$

lemma *the-part-some-elem* [*simp*]:
assumes $X \in P$
shows *the-part* ($\text{some-elem } X$) = X
 $\langle \text{proof} \rangle$

lemma *rep-comp*: $\llbracket X \in P; Y \in P \rrbracket \implies \text{rep-comp } (\text{some-elem } X) (\text{some-elem } Y) = \text{some-elem } (X \cdot Y)$
 $\langle \text{proof} \rangle$

lemma *Monoid-class-reps*:
assumes *Monoid* P $(\cdot)$ 1
shows *Monoid* ($\text{some-elem } ' P$) *rep-comp* ($\text{some-elem } 1$)
 $\langle \text{proof} \rangle$

lemma *Group-class-reps*:
assumes *Group* P $(\cdot)$ 1
shows *Group* ($\text{some-elem } ' P$) *rep-comp* ($\text{some-elem } 1$)
 $\langle \text{proof} \rangle$

lemma *some-elem-inj-on*: *inj-on* $\text{some-elem } P$
 $\langle \text{proof} \rangle$

lemma *some-elem-bij-betw*: *bij-betw some-elem P (some-elem ‘ P)*
 ⟨*proof*⟩

lemma *Monoid-iso-class-reps*:
assumes *Monoid P (·) 1*
shows *Monoid-isomorphism (restrict some-elem P) P (·) 1 (some-elem ‘ P) rep-comp (some-elem 1)*
 ⟨*proof*⟩

lemma *group-iso-class-reps*:
assumes *Group P (·) 1*
shows *group-isomorphism (restrict some-elem P) P (·) 1 (some-elem ‘ P) rep-comp (some-elem 1)*
 ⟨*proof*⟩

lemma *abelian-group-class-reps*:
assumes *Abelian-Group P (·) 1*
shows *Abelian-Group (some-elem ‘ P) rep-comp (some-elem 1)*
 ⟨*proof*⟩

end

context *normal-subgroup* **begin**

interpretation *reps: class-representatives G Factor-Group quotient-composition Class 1*
 ⟨*proof*⟩

theorem *group-isomorphism-Factor-Group-class-reps*:
group-isomorphism
(restrict some-elem Factor-Group) Factor-Group quotient-composition (Class 1)
(some-elem ‘ Factor-Group) reps.rep-comp (some-elem (Class 1))
 ⟨*proof*⟩

lemma *card-class-reps*:
card (some-elem ‘ Factor-Group) = card Factor-Group
 ⟨*proof*⟩

lemma *class-reps-subset*: *some-elem ‘ Factor-Group $\subseteq$ G*
 ⟨*proof*⟩

lemma *abelian-class-reps*:
assumes *Abelian-Group Factor-Group quotient-composition (Class 1)*
shows *Abelian-Group (some-elem ‘ Factor-Group) reps.rep-comp (some-elem (Class 1))*
 1))
 ⟨*proof*⟩

end

1.11 Abstract type of monoids

lemma *trivial-Monoid*: *Monoid* {undefined} ($\lambda x y.$ undefined) undefined
⟨proof⟩

lemma *trivial-Monoid-invertible*:
Monoid.invertible {undefined} ($\lambda x y.$ undefined) undefined undefined
⟨proof⟩

lemma *trivial-Group*: *Group* {undefined} ($\lambda x y.$ undefined) undefined
⟨proof⟩

typedef 'a monoid = {(M::'a set,f,e). *Monoid* M f e}
morphisms *dest-monoid monoid*
⟨proof⟩

declare *dest-monoid-inverse* [simp]

definition *trivial-monoid* $\equiv$ *monoid* ({undefined}, ($\lambda x y.$ undefined), undefined)

definition *mcarrier* **where** *mcarrier* m $\equiv$ *fst* (*dest-monoid* m)

definition *mmult* **where** *mmult* m $\equiv$ *fst* (*snd* (*dest-monoid* m))

definition *munit* **where** *munit* m $\equiv$ *snd* (*snd* (*dest-monoid* m))

lemma *monoid-is-Monoid* [iff]: *Monoid* (*mcarrier* m) (*mmult* m) (*munit* m)
⟨proof⟩

lemma *mmult-assoc* [simp]:
 $\llbracket a \in \text{mcarrier } m; b \in \text{mcarrier } m; c \in \text{mcarrier } m \rrbracket \implies \text{mmult } m (\text{mmult } m a b)$
 $c = \text{mmult } m a (\text{mmult } m b c)$
⟨proof⟩

lemma *m-left-unit*: $a \in \text{mcarrier } m \implies \text{mmult } m (\text{munit } m) a = a$
⟨proof⟩

lemma *m-right-unit*: $a \in \text{mcarrier } m \implies \text{mmult } m a (\text{munit } m) = a$
⟨proof⟩

lemma (**in** *Monoid*) *mcarrier-monoid*[simp]:
mcarrier (*monoid* (M, ($\cdot$), **1**)) = M
⟨proof⟩

lemma (**in** *Monoid*) *mmult-monoid*[simp]:
mmult (*monoid* (M, ($\cdot$), **1**)) = ($\cdot$)
⟨proof⟩

```

lemma (in Monoid) munit-monoid[simp]:
  munit (monoid (M, (·), 1)) = 1
  ⟨proof⟩

lemma monoid-collapse [simp]: monoid (mcarrier m, mmult m, munit m) = m
  ⟨proof⟩

Allows reference to the current monoid space within the locale as a value

definition (in Monoid) Self ≡ monoid (M, (·), 1)

lemma (in Monoid) mcarrier-Self [simp]: mcarrier Self = M
  ⟨proof⟩

lemma (in Monoid) mmult-Self [simp]: mmult Self = (·)
  ⟨proof⟩

lemma (in Monoid) munit-Self [simp]: munit Self = 1
  ⟨proof⟩

end

```

2 Partial orders, lattices and complete lattices

```

theory Order-Theory
  imports Set-Theory
begin

```

This theory uses no arithmetic; suppress HOL's $+/-$ notation (as *Ring-Theory* does) so that it composes without ambiguity when imported alongside the ring/group developments, which rebind those tokens to their locale operations.

```

no-notation plus (infixl <+> 65)
no-notation minus (infixl <-> 65)
unbundle no uminus-syntax

```

Carrier-set order theory in the locale idiom of *Group-Theory*: a *Partial-Order* fixes a carrier L and an order relation $\sqsubseteq$ (a binary predicate, with the reflexive / antisymmetric / transitive axioms restricted to L). On top of this we build binary lattices (least upper and greatest lower bounds of pairs) and complete lattices (of arbitrary subsets). Unlike HOL-Algebra's record-based *weak-partial-order* over an equivalence, we fix the relation directly, which keeps the statements close to the type-class order they specialise.

2.1 Partial orders

```

locale Partial-Order =
  fixes L :: 'a set and le :: 'a ⇒ 'a ⇒ bool (infix <⊆> 50)

```


assumes *reflexive* [*intro*, *simp*]: $a \in L \implies a \sqsubseteq a$
and *antisymmetric*: $\llbracket a \in L; b \in L; a \sqsubseteq b; b \sqsubseteq a \rrbracket \implies a = b$
and *transitive* [*trans*]: $\llbracket a \in L; b \in L; c \in L; a \sqsubseteq b; b \sqsubseteq c \rrbracket \implies a \sqsubseteq c$
begin

2.1.1 Bounds

x is an upper bound of A (a subset of the carrier) if it dominates every element of A .

definition *upper-bound* :: $'a \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$
where *upper-bound* $x A \longleftrightarrow x \in L \wedge (\forall a \in A. a \sqsubseteq x)$

definition *lower-bound* :: $'a \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$
where *lower-bound* $x A \longleftrightarrow x \in L \wedge (\forall a \in A. x \sqsubseteq a)$

lemma *upper-boundI*:
 $\llbracket x \in L; \bigwedge a. a \in A \implies a \sqsubseteq x \rrbracket \implies \text{upper-bound } x A$
 $\langle \text{proof} \rangle$

lemma *upper-boundD*: $\llbracket \text{upper-bound } x A; a \in A \rrbracket \implies a \sqsubseteq x$
 $\langle \text{proof} \rangle$

lemma *upper-bound-closed*: $\text{upper-bound } x A \implies x \in L$
 $\langle \text{proof} \rangle$

lemma *lower-boundI*:
 $\llbracket x \in L; \bigwedge a. a \in A \implies x \sqsubseteq a \rrbracket \implies \text{lower-bound } x A$
 $\langle \text{proof} \rangle$

lemma *lower-boundD*: $\llbracket \text{lower-bound } x A; a \in A \rrbracket \implies x \sqsubseteq a$
 $\langle \text{proof} \rangle$

lemma *lower-bound-closed*: $\text{lower-bound } x A \implies x \in L$
 $\langle \text{proof} \rangle$

2.1.2 Suprema and infima

s is a least upper bound (supremum) of A : an upper bound below every upper bound.

definition *is-sup* :: $'a \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$
where *is-sup* $s A \longleftrightarrow \text{upper-bound } s A \wedge (\forall x. \text{upper-bound } x A \longrightarrow s \sqsubseteq x)$

definition *is-inf* :: $'a \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$
where *is-inf* $i A \longleftrightarrow \text{lower-bound } i A \wedge (\forall x. \text{lower-bound } x A \longrightarrow x \sqsubseteq i)$

lemma *is-supI*:
 $\llbracket \text{upper-bound } s A; \bigwedge x. \text{upper-bound } x A \implies s \sqsubseteq x \rrbracket \implies \text{is-sup } s A$
 $\langle \text{proof} \rangle$

lemma *is-sup-upper*: $is-sup\ s\ A \implies upper-bound\ s\ A$
 $\langle proof \rangle$

lemma *is-sup-least*: $\llbracket is-sup\ s\ A; upper-bound\ x\ A \rrbracket \implies s \sqsubseteq x$
 $\langle proof \rangle$

lemma *is-infI*:
 $\llbracket lower-bound\ i\ A; \bigwedge x. lower-bound\ x\ A \implies x \sqsubseteq i \rrbracket \implies is-inf\ i\ A$
 $\langle proof \rangle$

lemma *is-inf-lower*: $is-inf\ i\ A \implies lower-bound\ i\ A$
 $\langle proof \rangle$

lemma *is-inf-greatest*: $\llbracket is-inf\ i\ A; lower-bound\ x\ A \rrbracket \implies x \sqsubseteq i$
 $\langle proof \rangle$

Suprema and infima are unique when they exist (antisymmetry).

lemma *is-sup-unique*:
assumes $is-sup\ s\ A$ **and** $is-sup\ s'\ A$
shows $s = s'$
 $\langle proof \rangle$

lemma *is-inf-unique*:
assumes $is-inf\ i\ A$ **and** $is-inf\ i'\ A$
shows $i = i'$
 $\langle proof \rangle$

end

2.2 Lattices

A lattice is a partial order in which every pair of elements has a supremum (join) and an infimum (meet).

locale *Lattice* = *Partial-Order* +
assumes $ex-sup: \llbracket a \in L; b \in L \rrbracket \implies \exists s. is-sup\ s\ \{a, b\}$
and $ex-inf: \llbracket a \in L; b \in L \rrbracket \implies \exists i. is-inf\ i\ \{a, b\}$
begin

definition *join* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\sqcup$ 65)
where $a \sqcup b = (THE\ s. is-sup\ s\ \{a, b\})$

definition *meet* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\sqcap$ 70)
where $a \sqcap b = (THE\ i. is-inf\ i\ \{a, b\})$

lemma *is-sup-join*:
assumes $a \in L\ b \in L$ **shows** $is-sup\ (a \sqcup b)\ \{a, b\}$
 $\langle proof \rangle$

lemma *is-inf-meet*:

assumes $a \in L$ $b \in L$ **shows** $is-inf (a \sqcap b) \{a, b\}$
 $\langle proof \rangle$

lemma *join-closed* [*intro*, *simp*]: $\llbracket a \in L; b \in L \rrbracket \implies a \sqcup b \in L$
 $\langle proof \rangle$

lemma *meet-closed* [*intro*, *simp*]: $\llbracket a \in L; b \in L \rrbracket \implies a \sqcap b \in L$
 $\langle proof \rangle$

The join dominates its arguments and is least among common upper bounds.

lemma *join-upper1*: $\llbracket a \in L; b \in L \rrbracket \implies a \sqsubseteq a \sqcup b$
 $\langle proof \rangle$

lemma *join-upper2*: $\llbracket a \in L; b \in L \rrbracket \implies b \sqsubseteq a \sqcup b$
 $\langle proof \rangle$

lemma *join-least*: $\llbracket a \in L; b \in L; a \sqsubseteq c; b \sqsubseteq c; c \in L \rrbracket \implies a \sqcup b \sqsubseteq c$
 $\langle proof \rangle$

lemma *meet-lower1*: $\llbracket a \in L; b \in L \rrbracket \implies a \sqcap b \sqsubseteq a$
 $\langle proof \rangle$

lemma *meet-lower2*: $\llbracket a \in L; b \in L \rrbracket \implies a \sqcap b \sqsubseteq b$
 $\langle proof \rangle$

lemma *meet-greatest*: $\llbracket a \in L; b \in L; c \sqsubseteq a; c \sqsubseteq b; c \in L \rrbracket \implies c \sqsubseteq a \sqcap b$
 $\langle proof \rangle$

2.2.1 Lattice laws

lemma *join-commutative*:
assumes $a \in L$ $b \in L$ **shows** $a \sqcup b = b \sqcup a$
 $\langle proof \rangle$

lemma *meet-commutative*:
assumes $a \in L$ $b \in L$ **shows** $a \sqcap b = b \sqcap a$
 $\langle proof \rangle$

lemma *join-idempotent* [*simp*]: $a \in L \implies a \sqcup a = a$
 $\langle proof \rangle$

lemma *meet-idempotent* [*simp*]: $a \in L \implies a \sqcap a = a$
 $\langle proof \rangle$

Absorption laws.

lemma *join-absorb*: $\llbracket a \in L; b \in L \rrbracket \implies a \sqcup (a \sqcap b) = a$
 $\langle proof \rangle$

lemma *meet-absorb*: $\llbracket a \in L; b \in L \rrbracket \implies a \sqcap (a \sqcup b) = a$

$\langle proof \rangle$

The order is recovered from the join (equivalently the meet).

lemma *le-iff-join*:

assumes $a: a \in L$ **and** $b: b \in L$ **shows** $(a \sqsubseteq b) = (a \sqcup b = b)$

$\langle proof \rangle$

end

2.3 Complete lattices

A complete lattice: every subset of the carrier has a supremum and an infimum.

locale *Complete-Lattice* = *Partial-Order* +

assumes *ex-Sup*: $A \subseteq L \implies \exists s. \text{is-sup } s \ A$

and *ex-Inf*: $A \subseteq L \implies \exists i. \text{is-inf } i \ A$

begin

definition *Sup* :: $'a \text{ set} \Rightarrow 'a$ ($\langle \sqcup \rangle$)

where $\sqcup \ A = (\text{THE } s. \text{is-sup } s \ A)$

definition *Inf* :: $'a \text{ set} \Rightarrow 'a$ ($\langle \sqcap \rangle$)

where $\sqcap \ A = (\text{THE } i. \text{is-inf } i \ A)$

lemma *is-sup-Sup*: $A \subseteq L \implies \text{is-sup } (\sqcup \ A) \ A$

$\langle proof \rangle$

lemma *is-inf-Inf*: $A \subseteq L \implies \text{is-inf } (\sqcap \ A) \ A$

$\langle proof \rangle$

lemma *Sup-closed* [*intro*, *simp*]: $A \subseteq L \implies \sqcup \ A \in L$

$\langle proof \rangle$

lemma *Inf-closed* [*intro*, *simp*]: $A \subseteq L \implies \sqcap \ A \in L$

$\langle proof \rangle$

lemma *Sup-upper*: $\llbracket A \subseteq L; a \in A \rrbracket \implies a \sqsubseteq \sqcup \ A$

$\langle proof \rangle$

lemma *Sup-least*: $\llbracket A \subseteq L; \text{upper-bound } x \ A \rrbracket \implies \sqcup \ A \sqsubseteq x$

$\langle proof \rangle$

lemma *Inf-lower*: $\llbracket A \subseteq L; a \in A \rrbracket \implies \sqcap \ A \sqsubseteq a$

$\langle proof \rangle$

lemma *Inf-greatest*: $\llbracket A \subseteq L; \text{lower-bound } x \ A \rrbracket \implies x \sqsubseteq \sqcap \ A$

$\langle proof \rangle$

Top and bottom: the sup and inf of the whole carrier.

definition *top* :: $'a$ ($\langle \top \rangle$) **where** $\top = \sqcup \ L$

definition *bottom* :: 'a ($\perp$) **where** $\perp = \sqcap L$

lemma *top-closed* [intro, simp]: $\top \in L$ *<proof>*

lemma *bottom-closed* [intro, simp]: $\perp \in L$ *<proof>*

lemma *top-greatest*: $a \in L \implies a \sqsubseteq \top$
<proof>

lemma *bottom-least*: $a \in L \implies \perp \sqsubseteq a$
<proof>

A complete lattice is a lattice (binary sup/inf are the two-element cases).

sublocale *Lattice*

<proof>

end

2.4 Isotone maps

An isotone (order-preserving, monotone) map between two partial orders.

definition *isotone* :: 'a set $\Rightarrow$ ('a $\Rightarrow$ 'a $\Rightarrow$ bool) $\Rightarrow$ 'b set $\Rightarrow$ ('b $\Rightarrow$ 'b $\Rightarrow$ bool) $\Rightarrow$ ('a $\Rightarrow$ 'b) $\Rightarrow$ bool

where *isotone* $LX\ leX\ LY\ leY\ f \longleftrightarrow$
 $f \in LX \rightarrow LY \wedge (\forall x \in LX. \forall y \in LX. leX\ x\ y \longrightarrow leY\ (f\ x)\ (f\ y))$

lemma *isotoneI*:

$\llbracket f \in LX \rightarrow LY; \bigwedge x\ y. \llbracket x \in LX; y \in LX; leX\ x\ y \rrbracket \implies leY\ (f\ x)\ (f\ y) \rrbracket$
 $\implies isotone\ LX\ leX\ LY\ leY\ f$
<proof>

lemma *isotone-closed*: $\llbracket isotone\ LX\ leX\ LY\ leY\ f; x \in LX \rrbracket \implies f\ x \in LY$
<proof>

lemma *isotone-le*: $\llbracket isotone\ LX\ leX\ LY\ leY\ f; x \in LX; y \in LX; leX\ x\ y \rrbracket \implies leY\ (f\ x)\ (f\ y)$
<proof>

2.5 Galois connections

A (monotone) Galois connection between two partial orders X and Y : a lower adjoint $f : X \rightarrow Y$ and an upper adjoint $g : Y \rightarrow X$ linked by the defining adjunction $f\ x \sqsubseteq_Y y \longleftrightarrow x \sqsubseteq_X g\ y$. This is the abstract backbone of correspondences such as the Galois correspondence between subgroups and intermediate fields.

locale *Galois-Connection* =

X : *Partial-Order* $LX\ leX$ + Y : *Partial-Order* $LY\ leY$

for LX **and** leX (**infix** $\sqsubseteq_X$ 50) **and** LY **and** leY (**infix** $\sqsubseteq_Y$ 50) +

fixes $f :: 'a \Rightarrow 'b$ **and** $g :: 'b \Rightarrow 'a$

assumes *f-closed*: $f \in LX \rightarrow LY$
and *g-closed*: $g \in LY \rightarrow LX$
and *galois*: $\llbracket x \in LX; y \in LY \rrbracket \implies f x \sqsubseteq_Y y \iff x \sqsubseteq_X g y$
begin

lemma *f-in*: $x \in LX \implies f x \in LY$ *<proof>*

lemma *g-in*: $y \in LY \implies g y \in LX$ *<proof>*

Unit and counit of the adjunction: $x \sqsubseteq g (f x)$ and $f (g y) \sqsubseteq y$.

lemma *unit*: $x \in LX \implies x \sqsubseteq_X g (f x)$
<proof>

lemma *counit*: $y \in LY \implies f (g y) \sqsubseteq_Y y$
<proof>

Both adjoints are isotone.

lemma *lower-isotone*: *isotone* LX *leX* LY *leY* f
<proof>

lemma *upper-isotone*: *isotone* LY *leY* LX *leX* g
<proof>

The composite $g \circ f$ is a closure operator on X : expansive ($?x \in LX \implies ?x \sqsubseteq_X g (f ?x)$), isotone, and idempotent. Dually $f \circ g$ is a kernel (interior) operator on Y . The two *semi-inverse* identities below — $f (g (f x)) = f x$ and $g (f (g y)) = g y$ — underlie idempotence.

lemma *lower-semi-inverse*: $x \in LX \implies f (g (f x)) = f x$
<proof>

lemma *upper-semi-inverse*: $y \in LY \implies g (f (g y)) = g y$
<proof>

Idempotence of the closure operator $g \circ f$ on X .

lemma *closure-idempotent*: $x \in LX \implies g (f (g (f x))) = g (f x)$
<proof>

Idempotence of the kernel operator $f \circ g$ on Y .

lemma *kernel-idempotent*: $y \in LY \implies f (g (f (g y))) = f (g y)$
<proof>

end

end

3 The lattice of subgroups

theory *Subgroup-Lattice*

```

imports Group-Theory Order-Theory
begin

```

The subgroups of a group G , ordered by inclusion, form a complete lattice: the infimum of a family of subgroups is their intersection (taken together with G itself, so the empty family has infimum G), and the supremum is the subgroup generated by their union. This is the first concrete consumer of the abstract order theory of *Order-Theory*.

```

context Group
begin

```

3.1 Intersections of subgroups

The intersection of a nonempty family of subgroups is a subgroup.

```

lemma subgroup-Inter:
  assumes ne:  $\mathcal{A} \neq \{\}$  and sub:  $\bigwedge H. H \in \mathcal{A} \implies \text{Subgroup } H \ G \ (\cdot) \ 1$ 
  shows  $\text{Subgroup } (\bigcap \mathcal{A}) \ G \ (\cdot) \ 1$ 
  <proof>

```

3.2 The complete lattice of subgroups

Carrier: the set of all subgroups of G ; order: inclusion.

```

definition subgroups :: 'a set set
  where  $\text{subgroups} = \{H. \text{Subgroup } H \ G \ (\cdot) \ 1\}$ 

```

```

lemma subgroups-iff [simp]:  $H \in \text{subgroups} \longleftrightarrow \text{Subgroup } H \ G \ (\cdot) \ 1$ 
  <proof>

```

Inclusion is a partial order on the subgroups.

```

interpretation subgroup-order: Partial-Order subgroups ( $\subseteq$ )
  <proof>

```

Every family of subgroups has an infimum (its intersection with G) and a supremum (the subgroup generated by its union), so the subgroups form a complete lattice.

```

interpretation subgroup-lattice: Complete-Lattice subgroups ( $\subseteq$ )
  <proof>

```

The lattice operations are the familiar ones: meet is intersection, join is the subgroup generated by the union, top is G and bottom is the trivial subgroup $\{1\}$.

```

lemma subgroup-top:  $\text{subgroup-lattice.top} = G$ 
  <proof>

```

```

end

```

```

end
theory Monoid-Iso-Classes
  imports Group-Theory
begin

```

4 Isomorphism classes of set-based monoids

New-Algebra represents a monoid by its carrier, composition, and unit rather than by a record, and *'a monoid* is the type-level wrapper for a valid such triple. Quotienting it by isomorphism gives isomorphism classes.

Classes are formed once, at the level of monoids, rather than separately for each stronger structure. Nothing is lost for groups: *Monoid-isomorphism.target-Group* makes being a group a property of the isomorphism class, and *isomorphic-as-monoids-iff-groups* shows the two notions of isomorphism agree on groups. Where group-ness is needed it is imposed as a constraint, as in *monoid-iso-class-of-monoid-eq-iff-groups* and *monoid-iso-class-eq-trivial-iff* below.

definition *monoid-iso-rel* :: *'a monoid* $\Rightarrow$ *'a monoid* $\Rightarrow$ *bool*

where

monoid-iso-rel *m n* $\equiv$
 $(mcarrier\ m, mmult\ m, munit\ m) \cong_M (mcarrier\ n, mmult\ n, munit\ n)$

Isomorphism is an equivalence relation on the wrapped monoids, so the quotient below records exactly the invariant needed by Jordan–Hölder.

quotient-type *'a monoid-iso-class* = *'a monoid* / *monoid-iso-rel*

morphisms *Rep-monoid-iso* *Abs-monoid-iso*

<proof>

definition *monoid-iso-class-of* :: *'a monoid* $\Rightarrow$ *'a monoid-iso-class*

where *monoid-iso-class-of* *m* = *Abs-monoid-iso* *m*

lemma *monoid-iso-class-of-eq-iff*:

monoid-iso-class-of *m* = *monoid-iso-class-of* *n* $\longleftrightarrow$ *monoid-iso-rel* *m n*
<proof>

lemma *monoid-iso-class-of-eq-iff-isomorphic*:

monoid-iso-class-of *m* = *monoid-iso-class-of* *n* $\longleftrightarrow$
 $(mcarrier\ m, mmult\ m, munit\ m) \cong_M (mcarrier\ n, mmult\ n, munit\ n)$
<proof>

The bridge clients use: comparing classes of monoids built from explicit triples is comparing the triples up to isomorphism.

lemma *monoid-iso-class-of-monoid-eq-iff*:

fixes *T U* :: *'a set* $\times$ (*'a* $\Rightarrow$ *'a* $\Rightarrow$ *'a*) $\times$ *'a*
assumes *T*: *Monoid* (*fst* *T*) (*fst* (*snd* *T*)) (*snd* (*snd* *T*))
and *U*: *Monoid* (*fst* *U*) (*fst* (*snd* *U*)) (*snd* (*snd* *U*))
shows *monoid-iso-class-of* (*monoid* *T*) = *monoid-iso-class-of* (*monoid* *U*) $\longleftrightarrow$ *T*
 $\cong_M$ *U*

<proof>

For groups the same statement holds with group isomorphism, because the two notions agree there. This is the form the Jordan–Hölder development needs.

lemma *monoid-iso-class-of-monoid-eq-iff-groups*:

fixes $T\ U :: 'a\ set \times ('a \Rightarrow 'a \Rightarrow 'a) \times 'a$

assumes T : *Group* ($fst\ T$) ($fst\ (snd\ T)$) ($snd\ (snd\ T)$)

and U : *Group* ($fst\ U$) ($fst\ (snd\ U)$) ($snd\ (snd\ U)$)

shows $monoid-iso-class-of\ (monoid\ T) = monoid-iso-class-of\ (monoid\ U) \longleftrightarrow T \cong_G U$

<proof>

4.1 The trivial isomorphism class

A distinguished class for the one-element monoid lets later developments discard repetition factors without choosing a representative from each quotient. *trivial-monoid* is the canonical representative; every one-element group is isomorphic to it.

definition *trivial-monoid-iso-class* :: $'a\ monoid-iso-class$

where $trivial-monoid-iso-class = monoid-iso-class-of\ trivial-monoid$

lemma *monoid-iso-class-eq-trivial-iff*:

assumes G : *Group* G *composition* *unit*

shows $monoid-iso-class-of\ (monoid\ (G,\ composition,\ unit)) = trivial-monoid-iso-class \longleftrightarrow G = \{unit\}$

<proof>

end

theory *Normal-Series*

imports *Group-Theory*

begin

5 Finite normal series

The canonical one-step series has the trivial subgroup at index zero and the full carrier thereafter. Values beyond its declared length are immaterial.

definition *one-step-normal-series-term* ::

$'a\ set \Rightarrow 'a \Rightarrow nat \Rightarrow 'a\ set$

where

$one-step-normal-series-term\ G\ unit\ i = (if\ i = 0\ then\ \{unit\}\ else\ G)$

A finite normal series is a finite chain of subgroups beginning at the trivial subgroup and ending at the ambient group, with each term normal in the next one. The chain is indexed by the natural numbers from 0 through n ; repeated terms are permitted, so strictness can be added separately when defining composition series.

The definition is deliberately stated with carrier sets and the existing *normal-subgroup* locale. In particular, it does not introduce a second representation of quotient groups or subgroup chains.

```

locale normal-series =
  G: Group G (·) 1
  for G and composition (infixl ⟨·⟩ 70) and unit (⟨1⟩) +
  fixes H :: nat ⇒ 'a set and n :: nat
  assumes bottom: H 0 = {1}
  and top: H n = G
  and term-subgroup:  $\bigwedge i. i \leq n \implies \text{Subgroup } (H\ i) \ G\ (\cdot)\ 1$ 
  and normal-step:
     $\bigwedge i. i < n \implies \text{normal-subgroup } (H\ i) \ (H\ (\text{Suc } i))\ (\cdot)\ 1$ 
begin

```

The factor at index i is packaged as the carrier, operation, and unit of the quotient of $H\ (\text{Suc } i)$ by $H\ i$. The definition is total in the index; the series assumptions provide the group interpretation whenever $i < n$.

```

definition series-factor ::
  nat ⇒ ('a set set × ('a set ⇒ 'a set ⇒ 'a set) × 'a set)
where
  series-factor i =
    (normal-subgroup.Factor-Group (H i) (H (Suc i)) (·) 1,
     Monoid-congruence.quotient-composition
       (H (Suc i)) (·)
       (normal-subgroup.Congruence (H i) (H (Suc i)) (·) 1),
     Equivalence.Class (H (Suc i))
       (normal-subgroup.Congruence (H i) (H (Suc i)) (·) 1) 1)

```

The finite factor sequence is ordered by increasing index.

```

definition series-factors ::
  ('a set set × ('a set ⇒ 'a set ⇒ 'a set) × 'a set) list
where
  series-factors = List.map series-factor [0.. $n$ ]

```

```

lemma length-series-factors:
  length series-factors = n
  ⟨proof⟩

```

```

lemma series-factors-prefix:
  assumes k:  $k \leq n$ 
  shows take k series-factors = List.map series-factor [0.. $k$ ]
  ⟨proof⟩

```

```

lemma series-factors-nth:
  assumes i:  $i < n$ 
  shows series-factors ! i = series-factor i
  ⟨proof⟩

```

lemma *series-factor-group*:
assumes $i: i < n$
shows $\text{Group } (\text{fst } (\text{series-factor } i))$
 $(\text{fst } (\text{snd } (\text{series-factor } i))) (\text{snd } (\text{snd } (\text{series-factor } i)))$
 $\langle \text{proof} \rangle$

The adjacent terms form a subgroup and are ordered by inclusion.

lemma *normal-step-subgroup*:
assumes $i: i < n$
shows $\text{Subgroup } (H\ i) (H\ (\text{Suc } i)) (\cdot) \mathbf{1}$
 $\langle \text{proof} \rangle$

lemma *normal-step-subset*:
assumes $i: i < n$
shows $H\ i \subseteq H\ (\text{Suc } i)$
 $\langle \text{proof} \rangle$

lemma *series-term-subset*:
assumes $i: i \leq n$
shows $H\ i \subseteq G$
 $\langle \text{proof} \rangle$

Inclusion propagates through the entire finite chain.

lemma *normal-series-mono*:
fixes $i\ j :: \text{nat}$
assumes $ij: i \leq j$ **and** $jn: j \leq n$
shows $H\ i \subseteq H\ j$
 $\langle \text{proof} \rangle$

lemma *series-bottom-subset*:
assumes $i: i \leq n$
shows $H\ 0 \subseteq H\ i$
 $\langle \text{proof} \rangle$

Every term in a prefix is a subgroup of the prefix's top term.

lemma *series-prefix-subgroup*:
assumes $k: k \leq n$ **and** $i: i \leq k$
shows $\text{Subgroup } (H\ i) (H\ k) (\cdot) \mathbf{1}$
 $\langle \text{proof} \rangle$

The terms up to any index form a normal series for that term.

lemma *prefix-normal-series*:
assumes $k: k \leq n$
shows $\text{normal-series } (H\ k) (\cdot) \mathbf{1} H\ k$
 $\langle \text{proof} \rangle$

lemma *zero-length-implies-trivial*:
assumes $n = 0$
shows $G = \{\mathbf{1}\}$

<proof>

end

context *Group* **begin**

Every group has the canonical one-step normal series from its trivial subgroup to its full carrier. For the trivial group the two terms coincide, as repetitions are permitted in a normal series.

lemma *one-step-normal-series*:

normal-series G $(\cdot)$ **1** (*one-step-normal-series-term* G **1**) **1**

<proof>

end

end

6 Finite relative normal chains

theory *Normal-Chain*

imports *Normal-Series Monoid-Iso-Classes HOL-Library.Multiset*

begin

6.1 Isomorphism classes of normal-series factors

The factor API stores quotient groups as the same carrier–operation–unit triples used throughout *New-Algebra*. We turn the valid factors of an arbitrary normal series into isomorphism classes and then into a multiset. When used with a composition series, these are the corresponding composition-factor classes.

context *normal-series*

begin

definition *series-factor-class* :: $\text{nat} \Rightarrow 'a \text{ set monoid-iso-class}$

where

series-factor-class $i = \text{monoid-iso-class-of } (\text{monoid } (\text{series-factor } i))$

The class sequence is ordered by increasing factor index.

definition *series-factor-classes* :: $'a \text{ set monoid-iso-class list}$

where

series-factor-classes = $\text{List.map series-factor-class } [0..<n]$

definition *series-factor-multiset* :: $'a \text{ set monoid-iso-class multiset}$

where

series-factor-multiset = $\text{mset series-factor-classes}$

lemma *length-series-factor-classes*:

length series-factor-classes = n

$\langle \text{proof} \rangle$

lemma *series-factor-classes-prefix*:

assumes $k: k \leq n$

shows $\text{take } k \text{ series-factor-classes} = \text{List.map series-factor-class } [0..<k]$

$\langle \text{proof} \rangle$

lemma *series-factor-classes-nth*:

assumes $i: i < n$

shows $\text{series-factor-classes } ! i = \text{series-factor-class } i$

$\langle \text{proof} \rangle$

lemma *size-series-factor-multiset*:

$\text{size series-factor-multiset} = n$

$\langle \text{proof} \rangle$

lemma *series-factor-class-eq-iff*:

assumes $i: i < n$ **and** $j: j < n$

shows $\text{series-factor-class } i = \text{series-factor-class } j \longleftrightarrow$

$\text{series-factor } i \cong_G \text{series-factor } j$

$\langle \text{proof} \rangle$

end

6.2 Normal factors

A refinement of one step in a normal series starts at an intermediate subgroup rather than at the trivial subgroup. The following locale records exactly the structure shared by such relative chains: finitely many subgroups of one ambient group, with each term normal in its successor.

definition *normal-factor* ::

$'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow ('a \Rightarrow 'a \Rightarrow 'a) \Rightarrow 'a \Rightarrow$
 $('a \text{ set set} \times ('a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set}) \times 'a \text{ set})$

where

$\text{normal-factor } K \ H \ \text{composition unit} \equiv$
 $(\text{normal-subgroup.Factor-Group } K \ H \ \text{composition unit},$
 $\text{Monoid-congruence.quotient-composition } H \ \text{composition}$
 $(\text{normal-subgroup.Congruence } K \ H \ \text{composition unit}),$
 $\text{Equivalence.Class } H$
 $(\text{normal-subgroup.Congruence } K \ H \ \text{composition unit}) \ \text{unit})$

definition *normal-factor-class* ::

$'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow ('a \Rightarrow 'a \Rightarrow 'a) \Rightarrow 'a \Rightarrow$
 $'a \text{ set monoid-iso-class}$

where

$\text{normal-factor-class } K \ H \ \text{composition unit} =$
 $\text{monoid-iso-class-of } (\text{monoid } (\text{normal-factor } K \ H \ \text{composition unit}))$

A normal factor is a group whenever its denominator is normal in its numerator.

Equality of factor classes can therefore be discharged by an ordinary group isomorphism between the corresponding quotient triples.

lemma *normal-factor-group*:

assumes *normal-subgroup K H composition unit*
shows *Group (fst (normal-factor K H composition unit))*
 (fst (snd (normal-factor K H composition unit)))
 (snd (snd (normal-factor K H composition unit)))
 ⟨proof⟩

lemma *normal-factor-class-eq-iff*:

assumes *KH: normal-subgroup K H composition unit*
and *LM: normal-subgroup L M composition unit*
shows *normal-factor-class K H composition unit =*
 normal-factor-class L M composition unit $\longleftrightarrow$
 normal-factor K H composition unit $\cong_G$ normal-factor L M composition unit
 ⟨proof⟩

A normal factor is trivial precisely when its denominator and numerator coincide. The quotient-level argument is proved once in *Factor-Group-eq-singleton-iff*; the carrier lemma below exposes it through the packaged factor triple, independently of the representative chosen for the trivial isomorphism class.

lemma *normal-factor-carrier-eq-singleton-iff*:

assumes *N: normal-subgroup K H composition unit*
shows *fst (normal-factor K H composition unit) =*
 {snd (snd (normal-factor K H composition unit))} $\longleftrightarrow K = H$
 ⟨proof⟩

lemma *normal-factor-class-eq-trivial-iff*:

assumes *N: normal-subgroup K H composition unit*
shows *normal-factor-class K H composition unit = trivial-monoid-iso-class $\longleftrightarrow K$*
= H
 ⟨proof⟩

locale *normal-chain* =

G: Group G (·) 1
for *G* **and** *composition (infixl <·> 70)* **and** *unit (<1>)* +
fixes *C :: nat $\Rightarrow$ 'a set* **and** *n :: nat*
assumes *term-subgroup*:
 $\bigwedge i. i \leq n \implies \text{Subgroup } (C\ i)\ G\ (\cdot)\ 1$
and *normal-step*:
 $\bigwedge i. i < n \implies \text{normal-subgroup } (C\ i)\ (C\ (\text{Suc } i))\ (\cdot)\ 1$
begin

lemma *step-subset*:

assumes *i < n*
shows *C i $\subseteq$ C (Suc i)*
 ⟨proof⟩

lemma *chain-mono*:

```

fixes  $i\ j :: \text{nat}$ 
assumes  $ij: i \leq j$  and  $jn: j \leq n$ 
shows  $C\ i \subseteq C\ j$ 
 $\langle \text{proof} \rangle$ 

```

```

definition chain-factor-class ::  $\text{nat} \Rightarrow 'a\ \text{set monoid-iso-class}$ 
where
  chain-factor-class  $i \equiv$ 
    normal-factor-class  $(C\ i)\ (C\ (\text{Suc}\ i))\ (\cdot)\ \mathbf{1}$ 

```

```

lemma prefix-normal-chain:
assumes  $k \leq n$ 
shows normal-chain  $G\ (\cdot)\ \mathbf{1}\ C\ k$ 
 $\langle \text{proof} \rangle$ 

```

end

```

sublocale normal-series  $\subseteq$  chain: normal-chain  $G\ (\cdot)\ \mathbf{1}\ H\ n$ 
 $\langle \text{proof} \rangle$ 

```

6.3 Isomorphism classes of normal-series factors

We turn the valid factors of an arbitrary normal series into isomorphism classes and then into a multiset. When used with a composition series, these are the corresponding composition-factor classes.

```

context normal-series
begin

```

```

lemma series-factor-eq-normal-factor:
  series-factor  $i = \text{normal-factor}\ (H\ i)\ (H\ (\text{Suc}\ i))\ (\cdot)\ \mathbf{1}$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma series-factor-class-eq-normal-factor-class:
  series-factor-class  $i =$ 
    normal-factor-class  $(H\ i)\ (H\ (\text{Suc}\ i))\ (\cdot)\ \mathbf{1}$ 
 $\langle \text{proof} \rangle$ 

```

end

end

```

theory Group-Operations
imports Group-Theory HOL-Computational-Algebra.Primes
begin

```

7 Group Actions

This theory formalises group actions with respect to the locale-based group infrastructure of *Group-Theory*, following the same development as *GroupAction.thy*

(which uses the record-based groups of HOL-Algebra).

A group action of a group $(G, \cdot, 1)$ on a set S is given by a group homomorphism $\varphi : G \rightarrow \text{Sym}(S)$, where $\text{Sym}(S)$ is the symmetric group on S (i.e. the group of bijections $S \rightarrow S$).

7.1 Definition and Basic Properties

```

locale Group-Action =
  grp: Group  $G$   $(\cdot)$  1 + transformations  $S$ 
  for  $G$  and composition (infixl  $\langle \cdot \rangle$  70) and unit  $(\langle 1 \rangle)$  and  $\varphi$  and  $S$  +
  assumes hom-closed:  $g \in G \implies \varphi g \in \text{Sym}$ 
    and hom-compose:  $\llbracket g \in G; h \in G \rrbracket \implies \varphi (g \cdot h) = \text{compose } S (\varphi g) (\varphi h)$ 
    and hom-unit:  $\varphi 1 = \text{identity } S$ 

```

begin

```

sublocale coset-notation  $\langle \text{proof} \rangle$ 

```

The action of **1** has no effect.

lemma one-is-id [simp]:

```

  assumes  $s \in S$ 
  shows  $\varphi 1 s = s$ 
   $\langle \text{proof} \rangle$ 

```

The action maps S to itself.

lemma action-closed [intro, simp]:

```

  assumes  $g \in G$   $s \in S$ 
  shows  $\varphi g s \in S$ 
   $\langle \text{proof} \rangle$ 

```

The image of any group element is a bijection on S .

lemma images-are-bij:

```

  assumes  $g \in G$ 
  shows bij-betw  $(\varphi g) S S$ 
   $\langle \text{proof} \rangle$ 

```

The action respects group multiplication.

lemma action-mult:

```

  assumes  $g \in G$   $h \in G$   $s \in S$ 
  shows  $\varphi g (\varphi h s) = \varphi (g \cdot h) s$ 
   $\langle \text{proof} \rangle$ 

```

The action of $\text{grp.inverse } g$ reverts the action of g .

lemma group-inv-rel:

```

  assumes  $g \in G$   $t \in S$   $\varphi g t = s$ 
  shows  $\varphi (\text{grp.inverse } g) s = t$ 
   $\langle \text{proof} \rangle$ 

```


lemma *action-inverse*:
assumes $g \in G \ s \in S$
shows $\varphi \ (grp.inverse \ g) \ (\varphi \ g \ s) = s$
 $\langle proof \rangle$

7.2 Stabilizer

The stabilizer of a point $s \in S$ is the set of group elements that fix s under the action.

definition *stabilizer*
where $stabilizer \ s = \{g \in G. \ \varphi \ g \ s = s\}$

The stabilizer of any point in S is a subgroup of G .

theorem *stabilizer-subgroup*:
assumes $s \in S$
shows $Subgroup \ (stabilizer \ s) \ G \ (\cdot) \ 1$
 $\langle proof \rangle$

7.3 Fixed Points

The fixed points of a group action are those elements of S that are fixed by every group element. Equivalently, a point is fixed if and only if its stabilizer is the whole group.

definition *fixed-points*
where $fixed-points = \{s \in S. \ G \subseteq stabilizer \ s\}$

lemma *fixed-points-subset*: $fixed-points \subseteq S$
 $\langle proof \rangle$

lemma *fixed-point-char*:
assumes $s \in S$
shows $s \in fixed-points \longleftrightarrow (\forall g \in G. \ \varphi \ g \ s = s)$
 $\langle proof \rangle$

lemma *fixed-pointI*:
assumes $s \in S \ \bigwedge g. \ g \in G \implies \varphi \ g \ s = s$
shows $s \in fixed-points$
 $\langle proof \rangle$

lemma *fixed-pointD*:
assumes $s \in fixed-points \ g \in G$
shows $\varphi \ g \ s = s$
 $\langle proof \rangle$

A point is fixed if and only if its stabilizer is the whole group.

lemma *fixed-point-stabilizer*:

assumes $s \in S$
shows $s \in \text{fixed-points} \longleftrightarrow \text{stabilizer } s = G$
 $\langle \text{proof} \rangle$

7.4 Orbits

The orbit of a point $s \in S$ under the group action is the set of all images of s under group elements.

definition *orbit*
where $\text{orbit } s \equiv \{\varphi \ g \ s \mid g. \ g \in G\}$

lemma *orbit-eq*: $\text{orbit } s = (\lambda g. \ \varphi \ g \ s) \ ` \ G$
 $\langle \text{proof} \rangle$

lemma *orbit-subset*:
assumes $s \in S$
shows $\text{orbit } s \subseteq S$
 $\langle \text{proof} \rangle$

lemma *orbit-self*:
assumes $s \in S$
shows $s \in \text{orbit } s$
 $\langle \text{proof} \rangle$

lemma *orbit-mem-iff*:
assumes $s \in S$
shows $t \in \text{orbit } s \longleftrightarrow (\exists g \in G. \ \varphi \ g \ s = t)$
 $\langle \text{proof} \rangle$

lemma *orbit-memI*:
assumes $g \in G$
shows $\varphi \ g \ s \in \text{orbit } s$
 $\langle \text{proof} \rangle$

lemma *orbit-memE*:
assumes $t \in \text{orbit } s$
obtains g **where** $g \in G \ \varphi \ g \ s = t$
 $\langle \text{proof} \rangle$

lemma *orbit-nonempty*: $\text{orbit } s \neq \{\}$
 $\langle \text{proof} \rangle$

A point is fixed precisely when its orbit is the singleton containing it.

lemma *fixed-point-iff-orbit-singleton*:
assumes $s \in S$
shows $s \in \text{fixed-points} \longleftrightarrow \text{orbit } s = \{s\}$
 $\langle \text{proof} \rangle$

lemma *card-orbit-gt0I*: $\text{finite } G \implies \text{card } (\text{orbit } s) > 0$

<proof>

7.5 The Orbit-Stabilizer Theorem

Two group elements give the same image under the action if and only if they lie in the same right coset of the stabilizer.

lemma *same-orbit-iff-same-coset*:

assumes $s \in S$ $g \in G$ $h \in G$

shows $\varphi g s = \varphi h s \iff \text{grp.inverse } g \cdot h \in \text{stabilizer } s$

<proof>

The orbit-stabilizer theorem. The map $\lambda g. \varphi g s$ from G to $\text{orbit } s$ is surjective, and its fibers are exactly the left cosets of the stabilizer. Hence the number of distinct images equals the index of the stabilizer, and Lagrange's theorem gives the cardinality form.

Every left coset of the stabilizer has the same cardinality as the stabilizer itself.

lemma *coset-fiber*:

assumes $s \in S$ **and** $g \in G$

shows $\{h \in G. \varphi h s = \varphi g s\} = g \cdot | \text{stabilizer } s$

<proof>

The orbit-stabilizer theorem (bijection form). The map that sends each orbit element to its fiber under $\lambda g. \varphi g s$ is a bijection from $\text{orbit } s$ to the set of left cosets of the stabilizer.

theorem *orbit-stabilizer-bij*:

assumes $s \in S$

shows $\text{bij-betw } (\lambda t. \{g \in G. \varphi g s = t\}) (\text{orbit } s) ((\lambda g. g \cdot | \text{stabilizer } s) \cdot G)$

<proof>

The orbit-stabilizer theorem (cardinality form).

theorem *orbit-stabilizer-card*:

assumes $s \in S$ **and** $\text{fin: finite } G$

shows $\text{card } G = \text{card } (\text{orbit } s) * \text{card } (\text{stabilizer } s)$

<proof>

7.6 Connection to Transformation Groups

The image $\varphi \cdot G$ is a subgroup of the symmetric group on S , and therefore forms a transformation group. This connects the orbit defined here with the orbit equivalence relation of *transformation-group*.

lemma *image-is-subgroup*: *Subgroup* $(\varphi \cdot G)$ *Sym* $(\text{compose } S)$ $(\text{identity } S)$

<proof>

sublocale *tg*: *transformation-group* $\varphi \cdot G$ S

<proof>

lemma *orbit-eq-Class*:
assumes $s \in S$
shows $\text{orbit } s = \text{tg.orbit.Class } s$
 $\langle \text{proof} \rangle$

7.7 The permutation representation

A group action is, by definition, a homomorphism into the symmetric group on S . We make this explicit, identify its kernel with the set of elements acting trivially on every point, and deduce that a *faithful* action embeds the group as a subgroup of the symmetric group: the permutation representation. This is what turns an action on a finite set into a concrete subgroup of permutations (e.g. a Galois group acting on the roots of a polynomial becomes a subgroup of Sym of the root set).

The action, restricted to G for extensionality, is a group homomorphism into the symmetric group.

interpretation *action-hom*:
 $\text{group-homomorphism } \text{restrict } \varphi \ G \ G \ (\cdot) \ \mathbf{1} \ \text{Sym } \text{compose } S \ \text{identity } S$
 $\langle \text{proof} \rangle$

On S , the image of a group element is the identity exactly when it fixes every point.

lemma *phi-eq-identity-iff*:
assumes $g \in G$
shows $\varphi \ g = \text{identity } S \longleftrightarrow (\forall s \in S. \varphi \ g \ s = s)$
 $\langle \text{proof} \rangle$

The kernel of the permutation representation is the set of group elements that fix every point of S , i.e. the intersection of all the stabilizers.

lemma *action-kernel-eq*:
 $\text{group-homomorphism.Ker } (\text{restrict } \varphi \ G) \ G \ (\text{identity } S) = \{g \in G. \forall s \in S. \varphi \ g \ s = s\}$
 $\langle \text{proof} \rangle$

An action is *faithful* if only the unit acts trivially on all of S .

definition *faithful*
where $\text{faithful} \longleftrightarrow (\forall g \in G. (\forall s \in S. \varphi \ g \ s = s) \longrightarrow g = \mathbf{1})$

lemma *faithful-iff-kernel-trivial*:
 $\text{faithful} \longleftrightarrow \text{group-homomorphism.Ker } (\text{restrict } \varphi \ G) \ G \ (\text{identity } S) = \{\mathbf{1}\}$
 $\langle \text{proof} \rangle$

A faithful action is injective.

lemma *faithful-inj*:
assumes *faithful*
shows $\text{inj-on } (\text{restrict } \varphi \ G) \ G$

<proof>

The permutation representation of a faithful action is a group isomorphism onto its image, exhibiting G as a subgroup of the symmetric group on S .

theorem *faithful-action-embeds*:

assumes *faithful*

shows *group-isomorphism (restrict φ G) G ($\cdot$) $\mathbf{1}$ ($\varphi \circ G$) (compose S) (identity S)*
<proof>

7.8 Orbit Partition

The orbits of a group action partition S . Using the connection to *transformation-group*, these facts follow from the general theory of equivalence relations.

If t is in the orbit of s , then s is in the orbit of t .

lemma *orbit-symmetric*:

assumes $s \in S$ $t \in \text{orbit } s$

shows $s \in \text{orbit } t$

<proof>

If t is in the orbit of s , the two orbits are equal.

lemma *orbit-subset-mem*:

assumes $s \in S$ $t \in \text{orbit } s$

shows $\text{orbit } t \subseteq \text{orbit } s$

<proof>

lemma *orbit-eq-mem*:

assumes $s \in S$ $t \in \text{orbit } s$

shows $\text{orbit } t = \text{orbit } s$

<proof>

Two orbits are either equal or disjoint.

lemma *orbit-disjoint*:

assumes $s \in S$ $t \in S$ $\text{orbit } s \neq \text{orbit } t$

shows $\text{orbit } s \cap \text{orbit } t = \{\}$

<proof>

The set of all orbits.

definition *orbits* :: 'b set set

where $\text{orbits} = \text{orbit} \circ S$

lemma *orbits-coverI*:

assumes $s \in S$

shows $\text{orbit } s \in \text{orbits}$

<proof>

lemma *orbits-coverE*:

assumes $X \in \text{orbits}$

obtains s **where** $s \in S$ $X = \text{orbit } s$
 $\langle \text{proof} \rangle$

The orbits cover S .

lemma *orbits-cover*: $\bigcup \text{orbits} = S$
 $\langle \text{proof} \rangle$

The orbits are pairwise disjoint.

lemma *orbits-disjoint*: *disjoint orbits*
 $\langle \text{proof} \rangle$

The nontrivial orbits are the orbits containing more than one point.

definition *nontrivial-orbits* :: 'b set set
where *nontrivial-orbits* = $\{X \in \text{orbits}. \neg \text{is-singleton } X\}$

lemma *nontrivial-orbits-subset*: *nontrivial-orbits* $\subseteq$ *orbits*
 $\langle \text{proof} \rangle$

The nontrivial orbits cover exactly the points that are not fixed.

lemma *nontrivial-orbits-cover*:
 $\bigcup \text{nontrivial-orbits} = S - \text{fixed-points}$
 $\langle \text{proof} \rangle$

Cardinality of S as a sum over orbits.

lemma *card-S-sum-orbits*:
assumes *finite S*
shows $\text{card } S = (\sum X \in \text{orbits}. \text{card } X)$
 $\langle \text{proof} \rangle$

Splitting the orbit partition into fixed points and nontrivial orbits gives the fixed-point form of the class equation.

lemma *fixed-point-class-equation*:
assumes *fin: finite S*
shows $\text{card } S = \text{card fixed-points} + (\sum X \in \text{nontrivial-orbits}. \text{card } X)$
 $\langle \text{proof} \rangle$

If every nontrivial orbit has cardinality divisible by k , then the same is true of the set of non-fixed points. This is the divisibility form of the fixed-point congruence.

lemma *dvd-card-nonfixed-points*:
assumes *fin: finite S*
and *dvd: $\bigwedge s. \llbracket s \in S; s \notin \text{fixed-points} \rrbracket \implies k \text{ dvd card } (\text{orbit } s)$*
shows $k \text{ dvd card } (S - \text{fixed-points})$
 $\langle \text{proof} \rangle$

If k divides the cardinality of every orbit, then k divides $\text{card } S$.

lemma *dvd-orbits-dvd-S*:

assumes $\text{finite } S \wedge X. X \in \text{orbits} \implies k \text{ dvd card } X$
shows $k \text{ dvd card } S$
 $\langle \text{proof} \rangle$

end

8 Sylow's Theorem

context *Monoid*
begin

definition $\text{power} :: 'a \Rightarrow \text{nat} \Rightarrow 'a$
where $\text{power } a \ n \equiv (((\cdot) \ a)^\wedge n) \ 1$

definition $\text{element-order} :: 'a \Rightarrow \text{nat}$
where $\text{element-order } a \equiv (\text{LEAST } n. n > 0 \wedge \text{power } a \ n = 1)$

lemma power-0 [simp]: $\text{power } a \ 0 = 1$
 $\langle \text{proof} \rangle$

lemma power-Suc [simp]: $\text{power } a \ (\text{Suc } n) = a \cdot \text{power } a \ n$
 $\langle \text{proof} \rangle$

lemma power-1 [simp]: $a \in M \implies \text{power } a \ 1 = a$
 $\langle \text{proof} \rangle$

lemma power-closed [intro, simp]: $a \in M \implies \text{power } a \ n \in M$
 $\langle \text{proof} \rangle$

lemma power-add : $a \in M \implies \text{power } a \ (m + n) = \text{power } a \ m \cdot \text{power } a \ n$
 $\langle \text{proof} \rangle$

lemma power-mult : $a \in M \implies \text{power } a \ (m * n) = \text{power } (\text{power } a \ m) \ n$
 $\langle \text{proof} \rangle$

lemma power-unit [simp]: $\text{power } 1 \ n = 1$
 $\langle \text{proof} \rangle$

lemma power-Suc-right : $a \in M \implies \text{power } a \ (\text{Suc } n) = \text{power } a \ n \cdot a$
 $\langle \text{proof} \rangle$

lemma power-commutes : $a \in M \implies a \cdot \text{power } a \ n = \text{power } a \ n \cdot a$
 $\langle \text{proof} \rangle$

lemma $\text{power-commutes-power}$:
 $a \in M \implies \text{power } a \ m \cdot \text{power } a \ n = \text{power } a \ n \cdot \text{power } a \ m$
 $\langle \text{proof} \rangle$

lemma *element-order-unit* [*simp*]: *element-order* **1** = 1
 ⟨*proof*⟩

end

context *Group* **begin**

lemma *power-inverse*:
assumes $g \in G$
shows $\text{power } (\text{inverse } g) \ n = \text{inverse } (\text{power } g \ n)$
 ⟨*proof*⟩

lemma *finite-power-order-exists*:
assumes *finite* $G \ g \in G$
shows $\exists n > 0. \text{power } g \ n = \mathbf{1}$
 ⟨*proof*⟩

lemma *element-order-pos*:
assumes *finite* $G \ g \in G$
shows *element-order* $g > 0$
 ⟨*proof*⟩

lemma *power-element-order*:
assumes *finite* $G \ g \in G$
shows $\text{power } g \ (\text{element-order } g) = \mathbf{1}$
 ⟨*proof*⟩

lemma *element-order-minimal*:
assumes *finite* $G \ g \in G \ n > 0 \ \text{power } g \ n = \mathbf{1}$
shows *element-order* $g \leq n$
 ⟨*proof*⟩

lemma *element-order-dvd*:
assumes *finite* $G \ g \in G \ \text{power } g \ n = \mathbf{1}$
shows *element-order* $g \ \text{dvd } n$
 ⟨*proof*⟩

lemma *power-mod-order*:
assumes *finite* $G \ g \in G$
shows $\text{power } g \ n = \text{power } g \ (n \bmod \text{element-order } g)$
 ⟨*proof*⟩

lemma *power-eq-order-False*:
assumes *finite* $G \ g \in G$
and $i < j \ j < \text{element-order } g$
and $\text{power } g \ i = \text{power } g \ j$
shows *False*
 ⟨*proof*⟩

lemma *power-inj-on-order*:


```

assumes finite  $G$   $g \in G$ 
shows inj-on (power  $g$ )  $\{.. $\text{element-order } g$ \}$ 
 $\langle \text{proof} \rangle$ 

```

end

8.1 Cyclic subgroups

The cyclic subgroup generated by an element g : the set of its powers. For g of finite order this is $(\cap) g \text{ ' } \{.. $\text{element-order } g$ \}$.

```

definition (in  $Group$ ) cyclic-subgroup :: ' $a \Rightarrow 'a$  set
  where cyclic-subgroup  $g = \text{power } g \text{ ' } \{.. $\text{element-order } g$ \}$ 

```

```

context  $Group$ 
begin

```

```

lemma cyclic-subgroup-eq-range:
  assumes fin: finite  $G$  and  $gG$ :  $g \in G$ 
  shows cyclic-subgroup  $g = \text{range } (\text{power } g)$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma cyclic-subgroup-subset:
  assumes  $g \in G$  shows cyclic-subgroup  $g \subseteq G$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma power-mem-cyclic-subgroup:
  assumes fin: finite  $G$  and  $gG$ :  $g \in G$ 
  shows power  $g$   $i \in \text{cyclic-subgroup } g$ 
 $\langle \text{proof} \rangle$ 

```

The cyclic subgroup is indeed a subgroup.

```

theorem cyclic-subgroup-is-subgroup:
  assumes fin: finite  $G$  and  $gG$ :  $g \in G$ 
  shows Subgroup (cyclic-subgroup  $g$ )  $G$   $(\cdot)$  1
 $\langle \text{proof} \rangle$ 

```

Its cardinality is the order of the generator.

```

theorem card-cyclic-subgroup:
  assumes fin: finite  $G$  and  $gG$ :  $g \in G$ 
  shows card (cyclic-subgroup  $g$ ) = element-order  $g$ 
 $\langle \text{proof} \rangle$ 

```

end

```

lemma (in  $Group$ ) element-order-dvd-card:
  assumes fin: finite  $G$  and  $gG$ :  $g \in G$ 
  shows element-order  $g$  dvd (card  $G$ )
 $\langle \text{proof} \rangle$ 

```

```

lemma (in Group) power-order-eq-1 [simp]:  $g \in G \implies \text{finite } G \implies \text{power } g (\text{card } G) = 1$ 
  <proof>

```

```

lemma (in Monoid-homomorphism) commutes-with-power:
  assumes  $g \in M$ 
  shows  $\eta (\text{source.power } g \ n) = \text{target.power } (\eta \ g) \ n$ 
  <proof>

```

```

lemma (in group-homomorphism) element-order-image-dvd:
  assumes  $\text{finite } G \ \text{finite } G' \ g \in G$ 
  shows  $\text{target.element-order } (\eta \ g) \ \text{dvd } \text{source.element-order } g$ 
  <proof>

```

```

lemma (in group-isomorphism) element-order-preserved:
  assumes  $\text{finite } G \ \text{finite } G' \ g \in G$ 
  shows  $\text{target.element-order } (\eta \ g) = \text{source.element-order } g$ 
  <proof>

```

The same fact stated with the raw *Monoid.element-order* applied to the explicit operations and units, rather than the locale-qualified *source.element-order* / *target.element-order*. This is the form a caller working with concrete groups (e.g. the symmetric group under $(\circ)/id$) can unify against directly — the prefixed versions are definitionally equal but do not match by name across interpretations.

```

lemma (in group-isomorphism) element-order-preserved':
  assumes  $\text{finite } G \ \text{finite } G' \ g \in G$ 
  shows  $\text{Monoid.element-order } (\cdot') \ (\mathbf{1}') \ (\eta \ g) = \text{Monoid.element-order } (\cdot) \ \mathbf{1} \ g$ 
  <proof>

```

```

lemma (in Group) element-order-power-prime:
  assumes fin:  $\text{finite } G$  and gG:  $g \in G$  and prime:  $\text{prime } p$  and dvd:  $p \ \text{dvd } \text{element-order } g$ 
  shows  $\text{element-order } (\text{power } g \ (\text{element-order } g \ \text{div } p)) = p$ 
  <proof>

```

```

end
theory Sylow-Theorems
  imports Group-Theory Group-Operations HOL-Computational-Algebra.Primes
begin

```

9 The First Sylow Theorem

```

context Group
begin

```

interpretation *left: left-translations-of-group* G $(\cdot)$ **1** $\langle proof \rangle$

The set of n -element subsets of G .

definition $\Omega :: nat \Rightarrow 'a\ set\ set$
where $\Omega\ n = \{B. B \subseteq G \wedge card\ B = n\}$

The left-translation action on $\Omega\ n$: each $g \in G$ acts by mapping a subset B to $(\cdot)\ g\ ' B = \{g \cdot b \mid b. b \in B\}$. We make the map extensional on $\Omega\ n$ so it lies in $\Omega\ n \rightarrow_E \Omega\ n$.

definition $\varphi\text{-left} :: nat \Rightarrow 'a \Rightarrow 'a\ set \Rightarrow 'a\ set$
where $\varphi\text{-left}\ n\ g = restrict\ (\lambda B. (\cdot)\ g\ ' B)\ (\Omega\ n)$

Membership in $\Omega\ n$.

lemma $\Omega\text{-memI}$ [*intro*]: $\llbracket B \subseteq G; card\ B = n \rrbracket \Longrightarrow B \in \Omega\ n$
 $\langle proof \rangle$

lemma $\Omega\text{-memD}$ [*dest*]: $B \in \Omega\ n \Longrightarrow B \subseteq G \wedge card\ B = n$
 $\langle proof \rangle$

The cardinality of $\Omega\ n$ is the binomial coefficient $card\ G\ choose\ n$.

lemma $card\text{-}\Omega$:
assumes $finite\ G$
shows $card\ (\Omega\ n) = card\ G\ choose\ n$
 $\langle proof \rangle$

Left multiplication by $g \in G$ is injective on the carrier.

lemma $left\text{-}mult\text{-}inj\text{-}on$:
assumes $g \in G$
shows $inj\text{-}on\ ((\cdot)\ g)\ G$
 $\langle proof \rangle$

Left multiplication maps subsets of G to subsets of G .

lemma $left\text{-}mult\text{-}image\text{-}subset$:
assumes $g \in G\ B \subseteq G$
shows $(\cdot)\ g\ ' B \subseteq G$
 $\langle proof \rangle$

Left multiplication preserves cardinality of subsets of G .

lemma $left\text{-}mult\text{-}image\text{-}card$:
assumes $g \in G\ B \subseteq G$
shows $card\ ((\cdot)\ g\ ' B) = card\ B$
 $\langle proof \rangle$

The action $\varphi\text{-left}\ n\ g$ maps $\Omega\ n$ to itself.

lemma $\varphi\text{-left}\text{-}closed$:
assumes $g \in G\ B \in \Omega\ n$
shows $(\cdot)\ g\ ' B \in \Omega\ n$

$\langle proof \rangle$

The action applied to an element of $\Omega\ n$.

lemma $\varphi\text{-left-apply}$ [simp]:
 assumes $B \in \Omega\ n$
 shows $\varphi\text{-left}\ n\ g\ B = (\cdot)\ g\ ` B$
 $\langle proof \rangle$

Outside $\Omega\ n$, the action is undefined.

lemma $\varphi\text{-left-undefined}$:
 assumes $B \notin \Omega\ n$
 shows $\varphi\text{-left}\ n\ g\ B = \text{undefined}$
 $\langle proof \rangle$

The action is extensional on $\Omega\ n$.

lemma $\varphi\text{-left-extensional}$: $\varphi\text{-left}\ n\ g \in \text{extensional}\ (\Omega\ n)$
 $\langle proof \rangle$

The action maps $\Omega\ n$ into itself.

lemma $\varphi\text{-left-mapsto}$:
 assumes $g \in G$
 shows $\varphi\text{-left}\ n\ g \in \Omega\ n \rightarrow_E \Omega\ n$
 $\langle proof \rangle$

Left multiplication by $g \in G$ followed by $\text{local.inverse}\ g$ is the identity on subsets of G .

lemma $\text{left-mult-image-inverse}$:
 assumes $g \in G\ B \subseteq G$
 shows $(\cdot)\ (\text{inverse}\ g)\ ` ((\cdot)\ g\ ` B) = B$
 $\langle proof \rangle$

The action $\varphi\text{-left}\ n\ g$ is a bijection on $\Omega\ n$.

lemma $\varphi\text{-left-bij}$:
 assumes $g \in G$
 shows $\text{bij-betw}\ (\varphi\text{-left}\ n\ g)\ (\Omega\ n)\ (\Omega\ n)$
 $\langle proof \rangle$

The action satisfies the unit law.

lemma $\varphi\text{-left-unit}$: $\varphi\text{-left}\ n\ \mathbf{1} = \text{identity}\ (\Omega\ n)$
 $\langle proof \rangle$

The action satisfies the composition law.

lemma $\varphi\text{-left-compose}$:
 assumes $g \in G\ h \in G$
 shows $\varphi\text{-left}\ n\ (g \cdot h) = \text{compose}\ (\Omega\ n)\ (\varphi\text{-left}\ n\ g)\ (\varphi\text{-left}\ n\ h)$
 $\langle proof \rangle$

The left-translation action on $\Omega\ n$ is a *Group-Action*.

theorem *subset-action*:
Group-Action $G \ (\cdot) \ \mathbf{1} \ (\varphi\text{-left } n) \ (\Omega \ n)$
 $\langle \text{proof} \rangle$

end

lemma (*in Group*) *Sylow-abelian*:
assumes *grp*: *Abelian-Group* $G \ (\cdot) \ \mathbf{1} \ \text{finite } G$
and *prime*: *prime* p **and** *dvd*: $p \ \text{dvd} \ \text{card } G$
shows $\exists g \in G. \ \text{element-order } g = p$
 $\langle \text{proof} \rangle$

9.1 Conjugation, Center, and Centralizer

context *Group* **begin**

The conjugation action: g acts on x by $g \ x \ g^{-1}$. We use $g \ x \ g^{-1}$ rather than $g^{-1} \ x \ g$ so that the map $g \mapsto (x \mapsto g \ x \ g^{-1})$ is a group homomorphism (not anti-homomorphism).

definition *conjugation* $:: 'a \Rightarrow 'a \Rightarrow 'a$
where *conjugation* $g \ x \equiv (\text{if } g \in G \wedge x \in G \text{ then } g \cdot x \cdot \text{inverse } g \text{ else } x)$

lemma *conjugation-closed*:
assumes $g \in G \ x \in G$
shows *conjugation* $g \ x \in G$
 $\langle \text{proof} \rangle$

lemma *conjugation-unit*:
assumes $x \in G$
shows *conjugation* $\mathbf{1} \ x = x$
 $\langle \text{proof} \rangle$

lemma *conjugation-compose*:
assumes $g \in G \ h \in G$
shows *conjugation* $(g \cdot h) \ x = \text{conjugation } g \ (\text{conjugation } h \ x)$
 $\langle \text{proof} \rangle$

lemma *conjugation-outside* [*simp*]:
 $x \notin G \implies \text{conjugation } g \ x = x$
 $\langle \text{proof} \rangle$

lemma *conjugation-bij*:
assumes $g \in G$
shows *bij-betw* $(\text{conjugation } g) \ G \ G$
 $\langle \text{proof} \rangle$

The restricted conjugation map, suitable for the *Group-Action* locale.

definition $\varphi\text{-conj} :: 'a \Rightarrow 'a \Rightarrow 'a$
where $\varphi\text{-conj } g = \text{restrict } (\text{conjugation } g) \ G$

lemma φ -conj-apply [simp]:
assumes $g \in G \ x \in G$
shows $\varphi\text{-conj } g \ x = g \cdot x \cdot \text{inverse } g$
 $\langle \text{proof} \rangle$

lemma φ -conj-outside [simp]:
 $x \notin G \implies \varphi\text{-conj } g \ x = \text{undefined}$
 $\langle \text{proof} \rangle$

lemma φ -conj-extensional: $\varphi\text{-conj } g \in \text{extensional } G$
 $\langle \text{proof} \rangle$

lemma φ -conj-mapsto:
assumes $g \in G$
shows $\varphi\text{-conj } g \in G \rightarrow_E G$
 $\langle \text{proof} \rangle$

lemma φ -conj-bij:
assumes $g \in G$
shows $\text{bij-betw } (\varphi\text{-conj } g) \ G \ G$
 $\langle \text{proof} \rangle$

lemma φ -conj-unit: $\varphi\text{-conj } \mathbf{1} = \text{identity } G$
 $\langle \text{proof} \rangle$

lemma φ -conj-compose:
assumes $g \in G \ h \in G$
shows $\varphi\text{-conj } (g \cdot h) = \text{compose } G \ (\varphi\text{-conj } g) \ (\varphi\text{-conj } h)$
 $\langle \text{proof} \rangle$

theorem conjugation-action: $\text{Group-Action } G \ (\cdot) \ \mathbf{1} \ \varphi\text{-conj } G$
 $\langle \text{proof} \rangle$

The center of G : elements that commute with everything.

definition $\text{center} :: 'a \text{ set}$
where $\text{center} = \{z \in G. \forall g \in G. z \cdot g = g \cdot z\}$

The centralizer of an element y : elements that commute with y .

definition $\text{centralizer} :: 'a \Rightarrow 'a \text{ set}$
where $\text{centralizer } y = \{g \in G. g \cdot y = y \cdot g\}$

lemma center-subset : $\text{center} \subseteq G$
 $\langle \text{proof} \rangle$

lemma center-mem-iff :
assumes $z \in G$
shows $z \in \text{center} \longleftrightarrow (\forall g \in G. z \cdot g = g \cdot z)$
 $\langle \text{proof} \rangle$

lemma *centralizer-subset*: $\text{centralizer } y \subseteq G$
 $\langle \text{proof} \rangle$

lemma *centralizer-mem-iff*:
assumes $g \in G \ y \in G$
shows $g \in \text{centralizer } y \longleftrightarrow g \cdot y = y \cdot g$
 $\langle \text{proof} \rangle$

Interpret the conjugation action to access orbits, stabilizers, and fixed points.

interpretation *conj*: *Group-Action* $G \ (\cdot) \ 1 \ \varphi\text{-conj } G$
 $\langle \text{proof} \rangle$

lemma *center-eq-fixed-points*: $\text{center} = \text{conj.fixed-points}$
 $\langle \text{proof} \rangle$

The class equation separates the singleton conjugacy classes, which are exactly the center, from the nontrivial conjugacy classes.

theorem *class-equation*:
assumes *finite* G
shows $\text{card } G = \text{card } \text{center} + (\sum X \in \text{conj.nontrivial-orbits}. \text{card } X)$
 $\langle \text{proof} \rangle$

lemma *centralizer-eq-stabilizer*:
assumes $y \in G$
shows $\text{centralizer } y = \text{conj.stabilizer } y$
 $\langle \text{proof} \rangle$

The center is a subgroup of G .

lemma *inverse-commutes*:
assumes $x \in G \ \forall g \in G. x \cdot g = g \cdot x$
shows $\forall g \in G. \text{inverse } x \cdot g = g \cdot \text{inverse } x$
 $\langle \text{proof} \rangle$

lemma *center-subgroup*: *Subgroup* $\text{center } G \ (\cdot) \ 1$
 $\langle \text{proof} \rangle$

The center is an abelian group.

lemma *center-abelian*: *Abelian-Group* $\text{center } (\cdot) \ 1$
 $\langle \text{proof} \rangle$

The centralizer of $y \in G$ is a subgroup of G .

lemma *centralizer-subgroup*:
assumes $y \in G$
shows *Subgroup* $(\text{centralizer } y) \ G \ (\cdot) \ 1$
 $\langle \text{proof} \rangle$

If y is not in the center, then its centralizer is a proper subgroup.

lemma *centralizer-proper*:
assumes $y \in G$ $y \notin \text{center } G$
shows $\text{card } (\text{centralizer } y) < \text{card } G$
 $\langle \text{proof} \rangle$

end

Preimage of a subgroup under a group homomorphism is a subgroup.

lemma (*in group-homomorphism*) *preimage-subgroup*:
assumes $\text{sub}: \text{Subgroup } S \text{ } G' \text{ } (\cdot)' \text{ } 1'$
defines $H \equiv \{g \in G. \eta \ g \in S\}$
shows $\text{Subgroup } H \text{ } G \text{ } (\cdot) \text{ } 1$
 $\langle \text{proof} \rangle$

Cardinality of the preimage: each coset in the subgroup contributes $\text{card } \text{Ker}$ elements.

lemma (*in group-homomorphism*) *preimage-card*:
assumes $\text{sub}: \text{Subgroup } S \text{ } G' \text{ } (\cdot)' \text{ } 1'$ **and** $\text{fin}: \text{finite } G$ **and** $\text{fin}': \text{finite } G'$
and $\text{img}: S \subseteq \eta^{-1} G$
defines $H \equiv \{g \in G. \eta \ g \in S\}$
shows $\text{card } H = \text{card } S * \text{card } \text{Ker}$
 $\langle \text{proof} \rangle$

9.2 First Sylow Theorem

theorem (*in Group*) *Sylow-I*:
fixes $p :: \text{nat}$ **and** $m :: \text{nat}$
assumes $\text{prime } p$
and $\text{finite } G$
and $p \wedge m \text{ dvd } \text{card } G$
shows $\exists H. \text{Subgroup } H \text{ } G \text{ } (\cdot) \text{ } 1 \wedge \text{card } H = p \wedge m$
 $\langle \text{proof} \rangle$

Cauchy's theorem for arbitrary finite groups is now an immediate corollary of the first Sylow theorem: taking $m = 1$ yields a subgroup of order p , and a group of prime order is cyclic, so any of its non-unit elements has order exactly p .)

corollary (*in Group*) *Cauchy*:
fixes $p :: \text{nat}$
assumes $\text{prime } p$ **and** $\text{fin}: \text{finite } G$ **and** $\text{dvd}: p \text{ dvd } \text{card } G$
shows $\exists g \in G. \text{element-order } g = p$
 $\langle \text{proof} \rangle$

10 The Second Sylow Theorem

If H is a subgroup of a group G and g is in G , then the conjugate $g H g^{-1}$ is again a subgroup of G .

context *Group* **begin**

definition *conjugate-subgroup* :: 'a $\Rightarrow$ 'a set $\Rightarrow$ 'a set
where *conjugate-subgroup* *g* *H* = *conjugation* *g* ' *H*

lemma *conjugate-subgroup-mem-iff*:
assumes *g* $\in$ *G* *H* $\subseteq$ *G* *x* $\in$ *G*
shows *x* $\in$ *conjugate-subgroup* *g* *H* $\longleftrightarrow$ ($\exists$ *h* $\in$ *H*. *x* = *g* $\cdot$ *h* $\cdot$ *inverse* *g*)
<proof>

lemma *conjugate-subgroup-subset*:
assumes *H* $\subseteq$ *G*
shows *conjugate-subgroup* *g* *H* $\subseteq$ *G*
<proof>

lemma *conjugate-subgroup-card*:
assumes *H* $\subseteq$ *G* *finite* *H*
shows *card* (*conjugate-subgroup* *g* *H*) = *card* *H*
<proof>

Conjugation is a homomorphism: it distributes over products and inverses.

lemma *conjugation-hom*:
assumes *h1* $\in$ *G* *h2* $\in$ *G*
shows *conjugation* *g* *h1* $\cdot$ *conjugation* *g* *h2* = *conjugation* *g* (*h1* $\cdot$ *h2*)
<proof>

lemma *conjugation-inverse*:
assumes *h* $\in$ *G*
shows *conjugation* *g* (*inverse* *h*) = *inverse* (*conjugation* *g* *h*)
<proof>

lemma *conjugate-subgroup-is-subgroup*:
assumes *sub*: *Subgroup* *H* *G* ($\cdot$) **1** **and** *g*: *g* $\in$ *G*
shows *Subgroup* (*conjugate-subgroup* *g* *H*) *G* ($\cdot$) **1**
<proof>

lemma *conjugate-subgroup-self*:
assumes *sub*: *Subgroup* *H* *G* ($\cdot$) **1** **and** *gH*: *g* $\in$ *H* **and** *fin*: *finite* *G*
shows *conjugate-subgroup* *g* *H* = *H*
<proof>

lemma *conjugate-subgroup-unit*:
assumes *H* $\subseteq$ *G*
shows *conjugate-subgroup* **1** *H* = *H*
<proof>

lemma *conjugate-subgroup-compose*:
assumes *g* $\in$ *G* *h* $\in$ *G* *H* $\subseteq$ *G*
shows *conjugate-subgroup* (*g* $\cdot$ *h*) *H* = *conjugate-subgroup* *g* (*conjugate-subgroup* *h* *H*)

$\langle proof \rangle$

If H normalises K (i.e. every element of H conjugates K to itself), then the set product $HK = H \times K$ is a subgroup of G .

definition *subgroup-product* :: 'a set $\Rightarrow$ 'a set $\Rightarrow$ 'a set
where *subgroup-product* $H K = (\lambda(h,k). h \cdot k) \text{ ' } (H \times K)$

lemma *subgroup-product-mem-iff*:
 $x \in \text{subgroup-product } H K \longleftrightarrow (\exists h \in H. \exists k \in K. x = h \cdot k)$
 $\langle proof \rangle$

lemma *subgroup-product-subset*:
assumes $H \subseteq G \ K \subseteq G$
shows *subgroup-product* $H K \subseteq G$
 $\langle proof \rangle$

lemma
assumes *Subgroup* $K G (\cdot) \mathbf{1} \ H \subseteq G$
shows *subgroup-product-superset-left*: $H \subseteq \text{subgroup-product } H K$
and *subgroup-product-superset-right*: $H \subseteq \text{subgroup-product } K H$
 $\langle proof \rangle$

lemma *normalizes-subgroup-product-is-subgroup*:
assumes *Hsub*: *Subgroup* $H G (\cdot) \mathbf{1}$
and *Ksub*: *Subgroup* $K G (\cdot) \mathbf{1}$
and *norm*: $\forall h \in H. \text{conjugate-subgroup } h K = K$
shows *Subgroup* $(\text{subgroup-product } H K) G (\cdot) \mathbf{1}$
 $\langle proof \rangle$

The normalizer of a subgroup H in G : the set of elements that normalize H under conjugation.

definition *normalizer* :: 'a set $\Rightarrow$ 'a set
where *normalizer* $H \equiv \{g \in G. \text{conjugate-subgroup } g H = H\}$

lemma *normalizer-subset*: *normalizer* $H \subseteq G$
 $\langle proof \rangle$

lemma *normalizer-mem-iff*:
 $g \in \text{normalizer } H \longleftrightarrow g \in G \wedge \text{conjugate-subgroup } g H = H$
 $\langle proof \rangle$

lemma *normalizer-subgroup*:
assumes *sub*: *Subgroup* $H G (\cdot) \mathbf{1}$ **and** *fin*: *finite* G
shows *Subgroup* $(\text{normalizer } H) G (\cdot) \mathbf{1}$
 $\langle proof \rangle$

lemma *subgroup-subset-normalizer*:
assumes *sub*: *Subgroup* $H G (\cdot) \mathbf{1}$ **and** *finite* G
shows $H \subseteq \text{normalizer } H$

<proof>

lemma *normal-in-normalizer*:

assumes *sub*: Subgroup H G $(\cdot)$ **1** **and** *fin*: finite G

shows normal-subgroup H (normalizer H) $(\cdot)$ **1**

<proof>

The order formula for a product whose factors normalize one another in the relevant direction. The second isomorphism theorem supplies the quotient cardinalities, so this bridge remains entirely within the native set-based API.

lemma *normalizes-subgroup-product-card*:

assumes *Hsub*: Subgroup H G $(\cdot)$ **1**

and *Ksub*: Subgroup K G $(\cdot)$ **1**

and *norm*: $\forall h \in H. \text{conjugate-subgroup } h \ K = K$

and *fin*: finite G

shows $\text{card} (\text{subgroup-product } H \ K) * \text{card} (H \cap K) = \text{card } H * \text{card } K$

<proof>

Sylow p -subgroups: subgroups whose order is the largest power of p dividing $\text{card } G$.

definition *Sylow-subgroups* :: $\text{nat} \Rightarrow 'a \text{ set set}$

where *Sylow-subgroups* $p =$

$\{H. \text{Subgroup } H \ G \ (\cdot) \ \mathbf{1} \wedge \text{card } H = p \wedge \text{multiplicity } p \ (\text{card } G)\}$

Sylow p -subgroups exist by the first Sylow theorem.

lemma *Sylow-subgroups-nonempty*:

assumes prime p finite G

shows *Sylow-subgroups* $p \neq \{\}$

<proof>

Conjugation preserves Sylow p -subgroups.

lemma *conjugate-Sylow-is-Sylow*:

assumes $P \in \text{Sylow-subgroups } p$ $g \in G$ finite G

shows *conjugate-subgroup* g $P \in \text{Sylow-subgroups } p$

<proof>

The conjugation action on subsets of G : each $g \in G$ maps a subset H to *conjugation* g $' H$. We restrict to *Sylow-subgroups* p to obtain a group action.

definition $\varphi\text{-Sylow}$:: $\text{nat} \Rightarrow 'a \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set}$

where $\varphi\text{-Sylow } p \ g = \text{restrict } (\text{conjugate-subgroup } g) \ (\text{Sylow-subgroups } p)$

lemma *Sylow-subgroups-subgroup*:

$P \in \text{Sylow-subgroups } p \implies \text{Subgroup } P \ G \ (\cdot) \ \mathbf{1}$

<proof>

lemma *Sylow-subgroups-subset*:

$P \in \text{Sylow-subgroups } p \implies P \subseteq G$

<proof>

lemma *Sylow-subgroups-card:*

$P \in \text{Sylow-subgroups } p \implies \text{card } P = p \wedge \text{multiplicity } p \text{ (card } G)$
 $\langle \text{proof} \rangle$

lemma φ -*Sylow-mapsto:*

assumes $g \in G$ *prime* p *finite* G
shows $\varphi\text{-Sylow } p \ g \in \text{Sylow-subgroups } p \rightarrow_E \text{Sylow-subgroups } p$
 $\langle \text{proof} \rangle$

lemma φ -*Sylow-bij:*

assumes $g: g \in G$ **and** $p: \text{prime } p$ **and** $\text{fin: finite } G$
shows $\text{bij-betw } (\varphi\text{-Sylow } p \ g) \ (\text{Sylow-subgroups } p) \ (\text{Sylow-subgroups } p)$
 $\langle \text{proof} \rangle$

lemma φ -*Sylow-unit:*

assumes *prime* p *finite* G
shows $\varphi\text{-Sylow } p \ \mathbf{1} = \text{identity } (\text{Sylow-subgroups } p)$
 $\langle \text{proof} \rangle$

lemma φ -*Sylow-compose:*

assumes $g: g \in G$ $h \in G$ *prime* p *finite* G
shows $\varphi\text{-Sylow } p \ (g \cdot h) = \text{compose } (\text{Sylow-subgroups } p) \ (\varphi\text{-Sylow } p \ g) \ (\varphi\text{-Sylow } p \ h)$
 $\langle \text{proof} \rangle$

lemma φ -*Sylow-action:*

assumes *prime* p *finite* G
shows $\text{Group-Action } G \ (\cdot) \ \mathbf{1} \ (\varphi\text{-Sylow } p) \ (\text{Sylow-subgroups } p)$
 $\langle \text{proof} \rangle$

lemma *prime-power-dvd-not-prime-dvd:*

fixes $p \ d :: \text{nat}$ **and** $k :: \text{nat}$
assumes *prime* p $d \ \text{dvd } p \wedge k \neg p \ \text{dvd } d$
shows $d = 1$
 $\langle \text{proof} \rangle$

The second Sylow theorem: any two Sylow p -subgroups of G are conjugate. That is, the conjugation action of G on $\text{Sylow-subgroups } p$ is transitive.

theorem *Sylow-II:*

assumes *prime* p *finite* G
and $P \in \text{Sylow-subgroups } p$ $Q \in \text{Sylow-subgroups } p$
shows $\exists g \in G. \text{conjugate-subgroup } g \ P = Q$
 $\langle \text{proof} \rangle$

A Sylow subgroup cannot normalize a distinct Sylow subgroup. The normalized product is a finite subgroup whose order is forced between the common Sylow order and the maximal p -power dividing $\text{card } G$.

lemma *normalizes-Sylow-subgroups-eq:*

assumes *prime*: prime p **and** *fin*: finite G
and P : $P \in \text{Sylow-subgroups } p$ **and** Q : $Q \in \text{Sylow-subgroups } p$
and *norm*: $\forall g \in P. \text{conjugate-subgroup } g \ Q = Q$
shows $Q = P$
 $\langle \text{proof} \rangle$

Conjugation is transitive on the Sylow p -subgroups.

lemma *Sylow-subgroups-orbit*:
assumes *prime*: prime p **and** *fin*: finite G
and P : $P \in \text{Sylow-subgroups } p$
shows $(\lambda g. \varphi\text{-Sylow } p \ g \ P) \text{ ` } G = \text{Sylow-subgroups } p$
 $\langle \text{proof} \rangle$

The number of Sylow p -subgroups divides the group order and is congruent to one modulo p .

theorem
assumes *prime*: prime p **and** *fin*: finite G
shows *Sylow-subgroups-card-dvd*: $\text{card } (\text{Sylow-subgroups } p) \text{ dvd } \text{card } G$
and *Sylow-III*: $p \text{ dvd } \text{card } (\text{Sylow-subgroups } p) - 1$
 $\langle \text{proof} \rangle$

end

end

11 Commutators and the Derived Subgroup

theory *Commutator*
imports *Sylow-Theorems*
begin

The commutator of two elements, the commutator subgroup of two subsets, and the derived (commutator) subgroup of a group. These underlie the derived series and the notion of a solvable group. (Custom notation is deferred; we use plain constant names while developing the theory.)

context *Group*
begin

The commutator of two elements.

definition *commutator-elt* :: $'a \Rightarrow 'a \Rightarrow 'a$
where *commutator-elt* $a \ b = a \cdot b \cdot \text{inverse } a \cdot \text{inverse } b$

Membership rule for commutators. Declared *simp* only: as an *intro* rule it back-chains on *every* carrier-membership goal $(?x \in G \text{ via } ?x = \text{commutator-elt } ?a \ ?b)$, which interacts badly with large introduction-rule sets and can loop.

lemma *commutator-elt-closed* [*simp*]:
 $\llbracket a \in G; b \in G \rrbracket \Longrightarrow \text{commutator-elt } a \ b \in G$

$\langle proof \rangle$

The commutator subgroup of two subsets of the carrier.

definition *commutator-subgroup* :: 'a set $\Rightarrow$ 'a set $\Rightarrow$ 'a set
where *commutator-subgroup* $H K = \langle \{ \text{commutator-elt } h k \mid h k. h \in H \wedge k \in K \} \rangle$

The derived subgroup of the whole group.

definition *derived* :: 'a set
where *derived* = *commutator-subgroup* $G G$

lemma *commutator-subgroup-is-subgroup*:
Subgroup (commutator-subgroup $H K$) $G (\cdot)$ 1
 $\langle proof \rangle$

lemma *commutator-subgroup-subset*:
commutator-subgroup $H K \subseteq G$
 $\langle proof \rangle$

lemma *derived-is-subgroup*: *Subgroup derived $G (\cdot)$ 1*
 $\langle proof \rangle$

lemma *derived-subset*: *derived $\subseteq G$*
 $\langle proof \rangle$

Generators of the commutator subgroup are commutators.

lemma *commutator-gen-mem*:
 $\llbracket h \in H; k \in K; H \subseteq G; K \subseteq G \rrbracket \Longrightarrow \text{commutator-elt } h k \in \text{commutator-subgroup } H K$
 $\langle proof \rangle$

Conjugation distributes over a commutator.

lemma *conjugation-commutator*:
assumes $g \in G \ a \in G \ b \in G$
shows *conjugation g (commutator-elt $a b$) = commutator-elt (conjugation $g a$) (conjugation $g b$)*
 $\langle proof \rangle$

The derived subgroup is normal.

lemma *derived-normal*: *normal-subgroup derived $G (\cdot)$ 1*
 $\langle proof \rangle$

The commutator subgroup is monotone in both arguments.

lemma *commutator-subgroup-mono*:
assumes $A \subseteq A' \ B \subseteq B'$
shows *commutator-subgroup $A B \subseteq \text{commutator-subgroup } A' B'$*
 $\langle proof \rangle$

A commutator computed inside a subgroup (with the restricted operation) equals the ambient commutator. A locality property of *commutator-elt*.

lemma *commutator-elt-subgroup-eq*:
assumes H : Subgroup H G $(\cdot)$ **1** **and** ab : $a \in H$ $b \in H$
shows $\text{Group.commutator-elt } H$ $(\cdot)$ **1** a $b = \text{commutator-elt } a$ b
 $\langle \text{proof} \rangle$

The commutator subgroup computed inside a subgroup is contained in the ambient one.

lemma *commutator-subgroup-subgroup-subset*:
assumes H : Subgroup H G $(\cdot)$ **1** **and** AB : $A \subseteq H$ $B \subseteq H$
shows $\text{Group.commutator-subgroup } H$ $(\cdot)$ **1** A $B \subseteq \text{commutator-subgroup } A$ B
 $\langle \text{proof} \rangle$

Generation is *absolute* for a subgroup: for $S \subseteq H$, the subgroup generated by S inside G coincides with the one generated inside H (the reverse inclusion to *generate-subgroup-subset*). The H -generated set is a subgroup of G containing S , so the minimal such (*generate*) is contained in it.

lemma *generate-subgroup-superset*:
assumes H : Subgroup H G $(\cdot)$ **1** **and** S : $S \subseteq H$
shows $\text{generate } S \subseteq \text{Group.generate } H$ $(\cdot)$ **1** S
 $\langle \text{proof} \rangle$

The commutator subgroup of subsets of H computed in G is contained in the one computed in H (the reverse of *commutator-subgroup-subgroup-subset*; the two are in fact equal).

lemma *commutator-subgroup-subset-subgroup*:
assumes H : Subgroup H G $(\cdot)$ **1** **and** AB : $A \subseteq H$ $B \subseteq H$
shows $\text{commutator-subgroup } A$ $B \subseteq \text{Group.commutator-subgroup } H$ $(\cdot)$ **1** A B
 $\langle \text{proof} \rangle$

11.1 The Derived Series and Solvable Groups

The derived series of a group is obtained by iterating the derived-subgroup construction. A group is solvable when its derived series reaches the trivial subgroup. As a first milestone we show that every abelian group is solvable.

The derived series, as a sequence of subsets of the carrier.

primrec $\text{derivedSeries} :: \text{nat} \Rightarrow 'a \text{ set}$
where
 $\text{derivedSeries } 0 = G$
 $| \text{derivedSeries } (\text{Suc } n) = \text{commutator-subgroup } (\text{derivedSeries } n) (\text{derivedSeries } n)$

A group is solvable if its derived series reaches the trivial subgroup.

definition $\text{solvable} :: \text{bool}$
where $\text{solvable} \equiv (\exists n. \text{derivedSeries } n = \{\mathbf{1}\})$

Each term of the derived series is a subgroup.

lemma *derivedSeries-subgroup*: Subgroup $(\text{derivedSeries } n)$ G $(\cdot)$ **1**

<proof>

lemma *derivedSeries-subset*: *derivedSeries* $n \subseteq G$ **and** *derivedSeries-unit-closed*: $\mathbf{1} \in \text{derivedSeries } n$

<proof>

The commutator subgroup of a subgroup H (with itself) is contained in H : the commutator generators lie in H (closed under the operation and inverse), and H , being a subgroup, contains the subgroup they generate.

lemma *commutator-subgroup-self-subset*:

assumes H : *Subgroup* H G $(\cdot)$ $\mathbf{1}$

shows *commutator-subgroup* H $H \subseteq H$

<proof>

The derived series is decreasing: *derivedSeries* $(\text{Suc } n) \subseteq \text{derivedSeries } n$.

lemma *derivedSeries-Suc-subset*: *derivedSeries* $(\text{Suc } n) \subseteq \text{derivedSeries } n$

<proof>

lemma *derivedSeries-antimono*: $m \leq n \implies \text{derivedSeries } n \subseteq \text{derivedSeries } m$

<proof>

In an abelian group every commutator is the unit.

lemma *commutator-elt-abelian*:

assumes *commutative-monoid* G $(\cdot)$ $\mathbf{1}$ **and** $a \in G$ $b \in G$

shows *commutator-elt* a $b = \mathbf{1}$

<proof>

In an abelian group the derived subgroup is trivial.

lemma *derived-trivial-if-abelian*:

assumes *commutative-monoid* G $(\cdot)$ $\mathbf{1}$

shows *derived* $= \{\mathbf{1}\}$

<proof>

Milestone: every abelian group is solvable.

theorem *abelian-imp-solvable*:

assumes *commutative-monoid* G $(\cdot)$ $\mathbf{1}$

shows *solvable*

<proof>

end

end

12 Transfer of Solvability

theory *Solvable-Transfer*

imports *Commutator*

begin

Solvability transfers along the basic group constructions: it passes to homomorphic (in particular quotient) images, and to subgroups. These are the load-bearing facts connecting a solvable group to its subquotients.

context *group-homomorphism*

begin

A homomorphism carries commutators to commutators.

lemma *image-commutator-elt*:

assumes $a \in G \ b \in G$

shows $\eta \ (source.commutator-elt \ a \ b) = target.commutator-elt \ (\eta \ a) \ (\eta \ b)$
 $\langle proof \rangle$

The image of a generated subgroup is contained in the subgroup generated by the image.

lemma *image-generate-sub*: $\eta \ ' \ source.generate \ S \subseteq target.generate \ (\eta \ ' \ (S \cap G))$

$\langle proof \rangle$

The reverse: the subgroup generated by the image is contained in the image of the generated subgroup. (No surjectivity needed.)

lemma *image-generate-super*: $target.generate \ (\eta \ ' \ (S \cap G)) \subseteq \eta \ ' \ source.generate \ S$

$\langle proof \rangle$

Hence the image of a commutator subgroup is the commutator subgroup of the images.

lemma *image-commutator-subgroup-sub*:

assumes $H \subseteq G \ K \subseteq G$

shows $\eta \ ' \ source.commutator-subgroup \ H \ K \subseteq target.commutator-subgroup \ (\eta \ ' \ H)$
 $(\eta \ ' \ K)$
 $\langle proof \rangle$

lemma *image-commutator-subgroup-super*:

assumes $H \subseteq G \ K \subseteq G$

shows $target.commutator-subgroup \ (\eta \ ' \ H) \ (\eta \ ' \ K) \subseteq \eta \ ' \ source.commutator-subgroup \ H \ K$
 $\langle proof \rangle$

lemma *image-commutator-subgroup-eq*:

assumes $H \subseteq G \ K \subseteq G$

shows $\eta \ ' \ source.commutator-subgroup \ H \ K = target.commutator-subgroup \ (\eta \ ' \ H)$
 $(\eta \ ' \ K)$
 $\langle proof \rangle$

For a surjective homomorphism, the derived series of the target is the image of the derived series of the source.

lemma *image-derivedSeries*:

assumes $\eta \text{ ' } G = G'$
shows $\eta \text{ ' } source.derivedSeries\ n = target.derivedSeries\ n$
 $\langle proof \rangle$

A surjective homomorphic image of a solvable group is solvable.

lemma *solvable-image*:
assumes $\eta \text{ ' } G = G'$ **and** *source.solvable*
shows *target.solvable*
 $\langle proof \rangle$

Unconditionally (no surjectivity), the image of the source derived series is contained in the target derived series.

lemma *image-derivedSeries-subset*: $\eta \text{ ' } source.derivedSeries\ n \subseteq target.derivedSeries\ n$
 $\langle proof \rangle$

A group admitting an injective homomorphism into a solvable group is solvable.
 If the target is solvable, its derived series reaches $\{1\}$ at some step n ; the image of the source derived series is contained there (*image-derivedSeries-subset*), so η maps *source.derivedSeries* n to $1'$, and injectivity (with $\eta\ 1 = 1'$) forces it to be $\{1\}$.

lemma *solvable-of-injective*:
assumes *inj*: *inj-on* $\eta\ G$ **and** *sol*: *target.solvable*
shows *source.solvable*
 $\langle proof \rangle$

end

A surjective homomorphic image of a solvable group is solvable (top level).

theorem (*in group-epimorphism*) *solvable-image-epi*:
assumes *source.solvable*
shows *target.solvable*
 $\langle proof \rangle$

12.1 Subgroups

We work in the ambient *Group* and refer to a subgroup's own group operations through the locale-qualified constants *Group.commutator-elt* $H\ (\cdot)\ 1$, *Group.generate* $H\ (\cdot)\ 1$ and *Group.derivedSeries* $H\ (\cdot)\ 1$. This bridge avoids the awkward parameter naming of the bare *Subgroup* locale and is reusable for any subgroup.

context *Group*
begin

The subgroup's derived series is contained in the ambient derived series.

lemma *derivedSeries-subgroup-subset*:
assumes H : *Subgroup* $H\ G\ (\cdot)\ 1$

shows $\text{Group.derivedSeries } H \ (\cdot) \ \mathbf{1} \ n \subseteq \text{derivedSeries } n$
 $\langle \text{proof} \rangle$

A subgroup of a solvable group is solvable.

theorem *solvable-subgroup*:
assumes H : *Subgroup* $H \ G \ (\cdot) \ \mathbf{1}$ **and** solv : *solvable*
shows $\text{Group.solvable } H \ (\cdot) \ \mathbf{1}$
 $\langle \text{proof} \rangle$

Shift lemma. Once the derived series has entered a subgroup H (at step k), the later terms are dominated by H 's own derived series: $\text{derivedSeries } (k + i)$ is contained in $H^{(i)}$. (Induction on i , using commutator monotonicity and that a commutator of subsets of H is computed inside H .)

lemma *derivedSeries-shift-subgroup*:
assumes H : *Subgroup* $H \ G \ (\cdot) \ \mathbf{1}$ **and** k : $\text{derivedSeries } k \subseteq H$
shows $\text{derivedSeries } (k + i) \subseteq \text{Group.derivedSeries } H \ (\cdot) \ \mathbf{1} \ i$
 $\langle \text{proof} \rangle$

end

12.2 Solvability of an extension

An extension of a solvable group by a solvable group is solvable. For a surjective homomorphism η : if the kernel and the image G' are both solvable, so is G . The derived series of G maps onto that of G' (*image-derivedSeries*), which reaches $\{\mathbf{1}'\}$; so $\text{source.derivedSeries } k \subseteq \text{Ker}$, and the kernel's own solvability then drives the remaining terms to $\{\mathbf{1}\}$ via the shift lemma.

theorem (*in group-epimorphism*) *solvable-extension*:
assumes solvKer : $\text{Group.solvable } \text{Ker } (\cdot) \ \mathbf{1}$
and solvImg : target.solvable
shows source.solvable
 $\langle \text{proof} \rangle$

end

13 The external direct product of two groups

theory *Group-Product*
imports *Solvable-Transfer*
begin

The external direct product of two groups: carrier the Cartesian product of the carriers, with componentwise operation and unit. This is the group-theoretic prerequisite for combining two solvable Galois groups (the compositum of two splitting fields embeds into the product of their Galois groups); its headline consequence is that a *product of solvable groups is solvable*.

locale *group-product* =

G : Group G $(\cdot)$ $\mathbf{1}$ + G' : Group G' $(\cdot')$ $\mathbf{1}'$
for G **and** composition (infixl $\langle \cdot \rangle$ 70) **and** unit $(\langle \mathbf{1} \rangle)$
and G' **and** composition' (infixl $\langle \cdot' \rangle$ 70) **and** unit' $(\langle \mathbf{1}' \rangle)$
begin

definition $pcarrier :: ('a \times 'b)$ set **where** $pcarrier = G \times G'$

definition $pcomp :: 'a \times 'b \Rightarrow 'a \times 'b$ (infixl $\langle \otimes_{\times} \rangle$ 70)
where $x \otimes_{\times} y = (fst\ x \cdot fst\ y, snd\ x \cdot' snd\ y)$

definition $punit :: 'a \times 'b$ ($\langle \mathbf{1}_{\times} \rangle$) **where** $\mathbf{1}_{\times} = (\mathbf{1}, \mathbf{1}')$

lemma $pcomp$ -closed [intro, simp]:
 $x \in pcarrrier \Longrightarrow y \in pcarrrier \Longrightarrow x \otimes_{\times} y \in pcarrrier$
 $\langle proof \rangle$

lemma $punit$ -closed [intro, simp]: $\mathbf{1}_{\times} \in pcarrrier$
 $\langle proof \rangle$

The direct product is a group.

theorem Group-product: Group $pcarrier$ $(\otimes_{\times})$ $\mathbf{1}_{\times}$
 $\langle proof \rangle$

sublocale prod: Group $pcarrier$ $(\otimes_{\times})$ $\mathbf{1}_{\times}$
 $\langle proof \rangle$

13.1 Projections are homomorphisms; the derived series is componentwise-bounded

The (extensional) first and second projections are group homomorphisms onto G and G' . We use *restrict fst pcarrrier* so the maps are extensional, as the homomorphism locale requires; on $pcarrier$ they agree with *fst* / *snd*.

definition $p1 :: 'a \times 'b \Rightarrow 'a$
where $p1 = restrict\ fst\ pcarrrier$

definition $p2 :: 'a \times 'b \Rightarrow 'b$
where $p2 = restrict\ snd\ pcarrrier$

lemma $p1$ -eq: $x \in pcarrrier \Longrightarrow p1\ x = fst\ x$
 $\langle proof \rangle$

lemma $p2$ -eq: $x \in pcarrrier \Longrightarrow p2\ x = snd\ x$
 $\langle proof \rangle$

lemma $p1$ -hom: group-homomorphism $p1\ pcarrrier$ $(\otimes_{\times})$ $\mathbf{1}_{\times}$ G $(\cdot)$ $\mathbf{1}$
 $\langle proof \rangle$

lemma $p2$ -hom: group-homomorphism $p2\ pcarrrier$ $(\otimes_{\times})$ $\mathbf{1}_{\times}$ G' $(\cdot')$ $\mathbf{1}'$

⟨*proof*⟩

An element of the product's derived series projects into each factor's derived series.

lemma *derivedSeries-p1*: $p1 \text{ ' } prod.derivedSeries\ n \subseteq G.derivedSeries\ n$
 ⟨*proof*⟩

lemma *derivedSeries-p2*: $p2 \text{ ' } prod.derivedSeries\ n \subseteq G'.derivedSeries\ n$
 ⟨*proof*⟩

13.2 Product of solvable groups is solvable

The direct product of two solvable groups is solvable. If both factors' derived series reach the trivial subgroup (at steps m, k), then at step $\max m\ k$ the product's derived series projects to $\{\mathbf{1}\}$ in each factor (the derived series is monotone decreasing), hence consists of the single element $\mathbf{1}_\times$.

theorem *solvable-product*:
assumes *solvG*: $G.solvable$ **and** *solvG'*: $G'.solvable$
shows *prod.solvable*
 ⟨*proof*⟩

end

end

14 The direct product of an indexed family of groups

theory *Group-Family-Product*
imports *Group-Theory*
begin

The external direct product of a family of groups $G\ i\ (i \in I)$: the carrier is the *extensional* dependent function space $Pi_E\ I\ G$ — functions defined on I with $x\ i \in G\ i$ for each index — with pointwise composition and unit. This generalises the binary external direct product of theory *Group-Product* to an arbitrary index set, mirroring *product-group* of *HOL-Algebra.Product-Groups* but in the locale idiom.

14.1 The product locale

A family of groups indexed by I : for each $i \in I$ a carrier $G\ i$ with composition $(\circ)\ i$ and unit $unit\ i$ forming a *comp-group*.

locale *group-family* =
fixes $I :: 'i\ set$

and $G :: 'i \Rightarrow 'a \text{ set}$
and $comp :: 'i \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
and $unit :: 'i \Rightarrow 'a$
assumes $comp\text{-}group: \bigwedge i. i \in I \implies Group (G\ i) (comp\ i) (unit\ i)$
begin

14.2 Carrier, composition and unit

The product carrier: extensional functions x on I with $x\ i \in G\ i$.

definition $Pcarrier :: ('i \Rightarrow 'a) \text{ set} (\langle \prod_G \rangle)$
where $\prod_G = (\Pi_E\ i \in I. G\ i)$

Pointwise composition (extensional on I).

definition $Pcomp :: ('i \Rightarrow 'a) \Rightarrow ('i \Rightarrow 'a) \Rightarrow ('i \Rightarrow 'a) (\text{infixl } \langle \otimes_\Pi \rangle\ 70)$
where $x \otimes_\Pi y = (\lambda i \in I. comp\ i\ (x\ i)\ (y\ i))$

The pointwise unit.

definition $Punit :: 'i \Rightarrow 'a (\langle 1_\Pi \rangle)$
where $1_\Pi = (\lambda i \in I. unit\ i)$

lemma $Pcarrier\text{-}memI$:

$\llbracket \bigwedge i. i \in I \implies x\ i \in G\ i; x \in \text{extensional } I \rrbracket \implies x \in \prod_G$
 $\langle proof \rangle$

lemma $Pcarrier\text{-}mem\text{-}component$: $\llbracket x \in \prod_G; i \in I \rrbracket \implies x\ i \in G\ i$
 $\langle proof \rangle$

lemma $Pcarrier\text{-}extensional$: $x \in \prod_G \implies x \in \text{extensional } I$
 $\langle proof \rangle$

lemma $Pcomp\text{-}closed$ $[intro, simp]$:

assumes $x \in \prod_G$ **and** $y \in \prod_G$ **shows** $x \otimes_\Pi y \in \prod_G$
 $\langle proof \rangle$

lemma $Punit\text{-}closed$ $[iff]$: $1_\Pi \in \prod_G$
 $\langle proof \rangle$

lemma $Pcomp\text{-}apply$: $i \in I \implies (x \otimes_\Pi y)\ i = comp\ i\ (x\ i)\ (y\ i)$
 $\langle proof \rangle$

lemma $Punit\text{-}apply$: $i \in I \implies 1_\Pi\ i = unit\ i$
 $\langle proof \rangle$

14.3 The product is a group

Two members of the product agree iff they agree componentwise on I (both being extensional).

lemma $Pcarrier\text{-}eqI$:

assumes $x \in \prod_G$ **and** $y \in \prod_G$ **and** $\bigwedge i. i \in I \implies x\ i = y\ i$
shows $x = y$
 $\langle proof \rangle$

theorem *Group-Pcarrier*: $Group\ \prod_G\ (\otimes_{\Pi})\ \mathbf{1}_{\Pi}$
 $\langle proof \rangle$

sublocale *product*: $Group\ Pcarrier\ (\otimes_{\Pi})\ Punit$
 $\langle proof \rangle$

14.4 Projections are homomorphisms

The projection onto the i th factor is a *comp-group* homomorphism from the product onto $G\ i$.

definition $Pproj :: 'i \Rightarrow ('i \Rightarrow 'a) \Rightarrow 'a$
where $Pproj\ i = restrict\ (\lambda x. x\ i)\ \prod_G$

lemma *Pproj-apply*: $x \in \prod_G \implies Pproj\ i\ x = x\ i$
 $\langle proof \rangle$

lemma *Pproj-hom*:
assumes $i: i \in I$
shows *group-homomorphism* $(Pproj\ i)\ \prod_G\ (\otimes_{\Pi})\ \mathbf{1}_{\Pi}\ (G\ i)\ (comp\ i)\ (unit\ i)$
 $\langle proof \rangle$

end

14.5 The product of a family of abelian groups is abelian

locale *abelian-group-family* = *group-family* +
assumes *abelian*: $\bigwedge i. i \in I \implies Abelian-Group\ (G\ i)\ (comp\ i)\ (unit\ i)$
begin

theorem *abelian-group-Pcarrier*: $Abelian-Group\ \prod_G\ (\otimes_{\Pi})\ \mathbf{1}_{\Pi}$
 $\langle proof \rangle$

end

end

theory *Zassenhaus-Lemma*
imports *Group-Theory*
begin

15 The Zassenhaus Lemma

Suppose that $H1$ is normal in H , and $K1$ is normal in K . The two wings of the butterfly are obtained by multiplying each normal subgroup by $H \cap K$; their denominators use the smaller intersections $H \cap K1$ and $K \cap H1$.

A locale records this symmetric configuration once. In particular, no ambient record updates or HOL-Algebra structures occur in the statement.

```
locale zassenhaus =
  G: Group G (·) 1 +
  H: subgroup-of-group H G (·) 1 +
  K: subgroup-of-group K G (·) 1 +
  H1: normal-subgroup H1 H (·) 1 +
  K1: normal-subgroup K1 K (·) 1
  for G H K H1 K1 and composition (infixl <·> 70) and unit (<1>)
begin
```

```
interpretation HK-sub: Subgroup H ∩ K G (·) 1
  <proof>
```

```
interpretation HK1-subG: Subgroup H ∩ K1 G (·) 1
  <proof>
```

```
interpretation HK1-sub: Subgroup H ∩ K1 H (·) 1
  <proof>
```

```
interpretation KH1-sub: Subgroup K ∩ H1 K (·) 1
  <proof>
```

```
interpretation HK1-normal: normal-subgroup H ∩ K1 H ∩ K (·) 1
  <proof>
```

```
interpretation KH1-normal: normal-subgroup K ∩ H1 H ∩ K (·) 1
  <proof>
```

```
interpretation middle-H: Subgroup H ∩ K H (·) 1
  <proof>
```

```
interpretation middle-K: Subgroup H ∩ K K (·) 1
  <proof>
```

```
interpretation middle-H-group: subgroup-of-group H ∩ K H (·) 1
  <proof>
```

```
interpretation middle-K-group: subgroup-of-group H ∩ K K (·) 1
  <proof>
```

```
interpretation HK1-group: subgroup-of-group H ∩ K1 H (·) 1
  <proof>
```


interpretation *KH1-group: subgroup-of-group* $K \cap H1 \ K \ (\cdot) \ 1$
 $\langle proof \rangle$

The four subgroup products making up the two wings.

definition *left-top where*
 $left-top \equiv (case-prod \ (\cdot)) \ ' \ ((H \cap K) \times H1)$

definition *left-bottom where*
 $left-bottom \equiv (case-prod \ (\cdot)) \ ' \ ((H \cap K1) \times H1)$

definition *right-top where*
 $right-top \equiv (case-prod \ (\cdot)) \ ' \ ((H \cap K) \times K1)$

definition *right-bottom where*
 $right-bottom \equiv (case-prod \ (\cdot)) \ ' \ ((K \cap H1) \times K1)$

interpretation *left-top-product: normal-subgroup-product* $H1 \ H \cap K \ H \ (\cdot) \ 1 \ \langle proof \rangle$

interpretation *left-bottom-product: normal-subgroup-product* $H1 \ H \cap K1 \ H \ (\cdot) \ 1$
 $\langle proof \rangle$

interpretation *right-top-product: normal-subgroup-product* $K1 \ H \cap K \ K \ (\cdot) \ 1 \ \langle proof \rangle$

interpretation *right-bottom-product: normal-subgroup-product* $K1 \ K \cap H1 \ K \ (\cdot) \ 1$
 $\langle proof \rangle$

lemma *left-top-eq [simp]: left-top = left-top-product.HK*
 $\langle proof \rangle$

lemma *left-bottom-eq [simp]: left-bottom = left-bottom-product.HK*
 $\langle proof \rangle$

lemma *right-top-eq [simp]: right-top = right-top-product.HK*
 $\langle proof \rangle$

lemma *right-bottom-eq [simp]: right-bottom = right-bottom-product.HK*
 $\langle proof \rangle$

lemma *left-top-subgroup: Subgroup left-top H (\cdot) 1*
 $\langle proof \rangle$

lemma *left-bottom-subgroup: Subgroup left-bottom H (\cdot) 1*
 $\langle proof \rangle$

lemma *right-top-subgroup: Subgroup right-top K (\cdot) 1*
 $\langle proof \rangle$

lemma *right-bottom-subgroup: Subgroup right-bottom K (\cdot) 1*
 $\langle proof \rangle$

lemma *left-bottom-subset-top*: $\text{left-bottom} \subseteq \text{left-top}$
 $\langle \text{proof} \rangle$

lemma *right-bottom-subset-top*: $\text{right-bottom} \subseteq \text{right-top}$
 $\langle \text{proof} \rangle$

lemma *left-top-memE*:
assumes $x \in \text{left-top}$
obtains $b \ n$ **where** $b \in H \cap K \ n \in H1 \ x = b \cdot n$
 $\langle \text{proof} \rangle$

lemma *left-bottom-memI*:
 $\llbracket c \in H \cap K1; n \in H1 \rrbracket \implies c \cdot n \in \text{left-bottom}$
 $\langle \text{proof} \rangle$

lemma *left-bottom-memE*:
assumes $x \in \text{left-bottom}$
obtains $c \ n$ **where** $c \in H \cap K1 \ n \in H1 \ x = c \cdot n$
 $\langle \text{proof} \rangle$

The lower subgroup in the left wing is normal in the upper one.

lemma *left-bottom-normal*: $\text{normal-subgroup left-bottom left-top } (\cdot) \ \mathbf{1}$
 $\langle \text{proof} \rangle$

The following two identities are the algebraic centre of the butterfly. An element of the middle intersection that is written as a product in one wing forces its normal-subgroup factor into the opposite subgroup.

lemma *middle-inter-left-bottom*:
 $(H \cap K) \cap \text{left-bottom} =$
 $(\text{case-prod } (\cdot)) \ ' ((H \cap K1) \times (K \cap H1))$
 $\langle \text{proof} \rangle$

interpretation *symmetric*: $\text{zassenhaus } G \ K \ H \ K1 \ H1 \ (\cdot) \ \mathbf{1} \ \langle \text{proof} \rangle$

lemma *middle-inter-right-bottom*:
 $(H \cap K) \cap \text{right-bottom} = (\text{case-prod } (\cdot)) \ ' ((K \cap H1) \times (H \cap K1))$
 $\langle \text{proof} \rangle$

The normality argument for the right wing is exactly the symmetric instance.

lemma *right-bottom-normal*: $\text{normal-subgroup right-bottom right-top } (\cdot) \ \mathbf{1}$
 $\langle \text{proof} \rangle$

15.1 The quotient isomorphism

interpretation *left-quotient*: $\text{normal-subgroup left-bottom left-top } (\cdot) \ \mathbf{1}$
 $\langle \text{proof} \rangle$

interpretation *right-quotient*: $\text{normal-subgroup right-bottom right-top } (\cdot) \ \mathbf{1}$
 $\langle \text{proof} \rangle$

lemma *middle-subset-left-top*: $H \cap K \subseteq \text{left-top}$
 $\langle \text{proof} \rangle$

lemma *middle-subset-right-top*: $H \cap K \subseteq \text{right-top}$
 $\langle \text{proof} \rangle$

lemma *middle-left-top-subgroup*: *Subgroup* $(H \cap K)$ *left-top* $(\cdot)$ **1**
 $\langle \text{proof} \rangle$

lemma *middle-right-top-subgroup*: *Subgroup* $(H \cap K)$ *right-top* $(\cdot)$ **1**
 $\langle \text{proof} \rangle$

interpretation *middle-left-top*: *subgroup-of-group* $H \cap K$ *left-top* $(\cdot)$ **1**
 $\langle \text{proof} \rangle$

interpretation *middle-right-top*: *subgroup-of-group* $H \cap K$ *right-top* $(\cdot)$ **1**
 $\langle \text{proof} \rangle$

interpretation *left-second*:
second-iso-theorem left-bottom $H \cap K$ *left-top* $(\cdot)$ **1** $\langle \text{proof} \rangle$

interpretation *right-second*:
second-iso-theorem right-bottom $H \cap K$ *right-top* $(\cdot)$ **1** $\langle \text{proof} \rangle$

lemma *left-second-product*: $\text{left-second.HK} = \text{left-top}$
 $\langle \text{proof} \rangle$

lemma *right-second-product*: $\text{right-second.HK} = \text{right-top}$
 $\langle \text{proof} \rangle$

lemma *left-second-factor-group*:
 $\text{left-second.Fract-HK-K} = \text{left-quotient.Factor-Group}$
 $\langle \text{proof} \rangle$

lemma *right-second-factor-group*:
 $\text{right-second.Fract-HK-K} = \text{right-quotient.Factor-Group}$
 $\langle \text{proof} \rangle$

lemma *common-denominator*:
 $(H \cap K) \cap \text{left-bottom} = (H \cap K) \cap \text{right-bottom}$
 $\langle \text{proof} \rangle$

interpretation *left-common*:
normal-subgroup $(H \cap K) \cap \text{left-bottom}$ $H \cap K$ $(\cdot)$ **1**
 $\langle \text{proof} \rangle$

interpretation *right-common*:
normal-subgroup $(H \cap K) \cap \text{right-bottom}$ $H \cap K$ $(\cdot)$ **1**

$\langle \text{proof} \rangle$

lemma *left-wing-isomorphism:*

(*left-quotient.Factor-Group*, *left-quotient.quotient-composition*,
 left-quotient.Class 1) $\cong_G$
(*left-second.Fract-H-HIntK*, *left-common.quotient-composition*,
 left-common.Class 1)
 $\langle \text{proof} \rangle$

lemma *right-wing-isomorphism:*

(*right-quotient.Factor-Group*, *right-quotient.quotient-composition*,
 right-quotient.Class 1) $\cong_G$
(*right-second.Fract-H-HIntK*, *right-common.quotient-composition*,
 right-common.Class 1)
 $\langle \text{proof} \rangle$

lemma *common-factor-group:*

(*left-second.Fract-H-HIntK*, *left-common.quotient-composition*,
 left-common.Class 1) =
(*right-second.Fract-H-HIntK*, *right-common.quotient-composition*,
 right-common.Class 1)
 $\langle \text{proof} \rangle$

The Zassenhaus lemma: the quotients represented by the two wings of the butterfly are isomorphic. The quotient operations and units are displayed explicitly, as is customary in New-Algebra's set-based API.

theorem *butterfly-lemma:*

(*left-quotient.Factor-Group*, *left-quotient.quotient-composition*,
 left-quotient.Class 1) $\cong_G$
(*right-quotient.Factor-Group*, *right-quotient.quotient-composition*,
 right-quotient.Class 1)
 $\langle \text{proof} \rangle$

end

end

16 The Schreier refinement theorem

theory *Schreier-Refinement*

imports *Normal-Chain Zassenhaus-Lemma*

begin

16.1 Refining pairs of finite normal series

The Zassenhaus lemma compares one refinement cell determined by a step from each of two normal series. We name the two families of subgroup products independently of a particular cell. Fixing one index produces a row of intermediate

subgroups refining the corresponding step of the other series.

```

locale normal-series-pair =
  A: normal-series G (·) 1 A m +
  B: normal-series G (·) 1 B n
  for G and composition (infixl ⟨·⟩ 70) and unit (⟨1⟩)
    and A :: nat ⇒ 'a set and m
    and B :: nat ⇒ 'a set and n
begin

definition left-refinement-term :: nat ⇒ nat ⇒ 'a set
  where left-refinement-term i j ≡ (case-prod (·)) ' ((A (Suc i) ∩ B j) × A i)

definition right-refinement-term :: nat ⇒ nat ⇒ 'a set
  where right-refinement-term j i ≡ (case-prod (·)) ' ((B (Suc j) ∩ A i) × B j)

end

locale series-refinement-cell =
  normal-series-pair G (·) 1 A m B n
  for G and composition (infixl ⟨·⟩ 70) and unit (⟨1⟩)
    and A :: nat ⇒ 'a set and m
    and B :: nat ⇒ 'a set and n
    and i j +
  assumes i-lt: i < m and j-lt: j < n
begin

lemma ambient-group:
  Group G (·) 1
  ⟨proof⟩

lemma A-step-top-subgroup:
  subgroup-of-group (A (Suc i)) G (·) 1
  ⟨proof⟩

lemma B-step-top-subgroup:
  subgroup-of-group (B (Suc j)) G (·) 1
  ⟨proof⟩

interpretation cell: zassenhaus
  G A (Suc i) B (Suc j) A i B j (·) 1
  ⟨proof⟩

lemma left-refinement-bottom-eq:
  left-refinement-term i j = cell.left-bottom
  ⟨proof⟩

lemma left-refinement-top-eq:
  left-refinement-term i (Suc j) = cell.left-top
  ⟨proof⟩

```

lemma *right-refinement-bottom-eq*:
 $\text{right-refinement-term } j \ i = \text{cell.right-bottom}$
 $\langle \text{proof} \rangle$

lemma *right-refinement-top-eq*:
 $\text{right-refinement-term } j \ (\text{Suc } i) = \text{cell.right-top}$
 $\langle \text{proof} \rangle$

lemma *left-refinement-normal*:
 $\text{normal-subgroup } (\text{left-refinement-term } i \ j) \ (\text{left-refinement-term } i \ (\text{Suc } j)) \ (\cdot) \ 1$
 $\langle \text{proof} \rangle$

lemma *right-refinement-normal*:
 $\text{normal-subgroup } (\text{right-refinement-term } j \ i) \ (\text{right-refinement-term } j \ (\text{Suc } i)) \ (\cdot) \ 1$
 $\langle \text{proof} \rangle$

Opposite sides of every refinement cell have the same native quotient-group isomorphism class. This is the factor-level form of Zassenhaus needed by a later Schreier refinement theorem; the statement contains no HOL-Algebra records and no representatives of quotient isomorphisms.

theorem *refinement-cell-factor-class-eq*:
 $\text{normal-factor-class } (\text{left-refinement-term } i \ j) \ (\text{left-refinement-term } i \ (\text{Suc } j)) \ (\cdot) \ 1 =$
 $\text{normal-factor-class } (\text{right-refinement-term } j \ i) \ (\text{right-refinement-term } j \ (\text{Suc } i)) \ (\cdot) \ 1$
 $\langle \text{proof} \rangle$

end

The factors of a refinement are read row by row. The following elementary lemmas turn a rectangular list of entries into finite sums of singleton multisets; this makes transposition of the refinement matrix explicit in the final proof.

lemma *mset-map-upt*:
 $\text{mset } (\text{List.map } f \ [0..<n]) = (\sum i < n. \ \{\#f \ i\# \})$
 $\langle \text{proof} \rangle$

lemma *mset-concat-map-upt*:
 $\text{mset } (\text{concat } (\text{List.map } (\lambda i. \ \text{List.map } (f \ i) \ [0..<n]) \ [0..<m])) = (\sum i < m. \ \sum j < n. \ \{\#f \ i \ j\# \})$
 $\langle \text{proof} \rangle$

lemma *length-concat-map-upt*:
 $\text{length } (\text{concat } (\text{List.map } (\lambda i. \ \text{List.map } (f \ i) \ [0..<n]) \ [0..<m])) = m * n$
 $\langle \text{proof} \rangle$

At the endpoints of a refinement row, one factor in a subgroup product is either trivial or the whole carrier. For intermediate terms, intersecting the carrier with another ambient subgroup and multiplying by a normal subgroup again gives an

ambient subgroup. These generic facts keep the row proofs below independent of quotient representations.

lemma *subgroup-product-with-trivial-left*:
assumes K : Subgroup K G composition unit
shows (case-prod composition) ‘ $(\{unit\} \times K) = K$
 $\langle proof \rangle$

lemma *subgroup-product-with-carrier-left*:
assumes K : Subgroup K H composition unit
shows (case-prod composition) ‘ $(H \times K) = H$
 $\langle proof \rangle$

lemma *intersection-normal-product-subgroup*:
assumes G : Group G composition unit
and N : normal-subgroup K H composition unit
and H : Subgroup H G composition unit
and L : Subgroup L G composition unit
shows Subgroup $((case-prod composition) ‘ ((H \cap L) \times K))$
 G composition unit
 $\langle proof \rangle$

context *normal-series-pair*
begin

For every pair of valid step indices, the assumptions of *series-refinement-cell* follow directly from the two series. We construct the locale predicate without interpreting it, so clients can reuse the proved cell API without re-elaborating the Zassenhaus locale.

lemma *series-refinement-cellI*:
assumes i : $i < m$ **and** j : $j < n$
shows series-refinement-cell G $(\cdot)$ **1** A m B n i j
 $\langle proof \rangle$

16.2 Refinement rows

The left row at index i runs from A i to A $(Suc\ i)$; the right row at index j runs from B j to B $(Suc\ j)$. Thus concatenating rows in increasing outer index refines the original series. The normal-chain theorems verify every intermediate subgroup and adjacent normality condition.

lemma *left-refinement-start*:
assumes i : $i < m$
shows left-refinement-term i $0 = A$ i
 $\langle proof \rangle$

lemma *left-refinement-end*:
assumes i : $i < m$
shows left-refinement-term i $n = A$ $(Suc\ i)$
 $\langle proof \rangle$

lemma *right-refinement-start*:
 assumes $j: j < n$
 shows *right-refinement-term* j $0 = B\ j$
 $\langle proof \rangle$

lemma *right-refinement-end*:
 assumes $j: j < n$
 shows *right-refinement-term* j $m = B\ (Suc\ j)$
 $\langle proof \rangle$

lemma *left-refinement-step*:
 assumes $i: i < m$ and $j: j < n$
 shows *normal-subgroup* (*left-refinement-term* $i\ j$) (*left-refinement-term* $i\ (Suc\ j)$) $(\cdot)$ **1**
 $\langle proof \rangle$

lemma *left-refinement-term-subgroup*:
 assumes $i: i < m$ and $j: j \leq n$
 shows *Subgroup* (*left-refinement-term* $i\ j$) $G\ (\cdot)$ **1**
 $\langle proof \rangle$

lemma *right-refinement-step*:
 assumes $i: i < m$ and $j: j < n$
 shows *normal-subgroup* (*right-refinement-term* $j\ i$) (*right-refinement-term* $j\ (Suc\ i)$) $(\cdot)$ **1**
 $\langle proof \rangle$

lemma *right-refinement-term-subgroup*:
 assumes $i: i \leq m$ and $j: j < n$
 shows *Subgroup* (*right-refinement-term* $j\ i$) $G\ (\cdot)$ **1**
 $\langle proof \rangle$

lemma *left-refinement-row*:
 assumes $i: i < m$
 shows *normal-chain* $G\ (\cdot)$ **1** (*left-refinement-term* i) n
 $\langle proof \rangle$

lemma *right-refinement-row*:
 assumes $j: j < n$
 shows *normal-chain* $G\ (\cdot)$ **1** (*right-refinement-term* j) m
 $\langle proof \rangle$

16.3 Factor-class refinements

The two lists below flatten the left and right refinement rows in row-major order. Each entry is the native isomorphism class of the corresponding normal-chain factor; repeated or trivial factors are retained.

definition *left-refinement-factor-class* :: $nat \Rightarrow nat \Rightarrow 'a$ set monoid-iso-class

where
 $\text{left-refinement-factor-class } i \ j \equiv$
 $\text{normal-factor-class } (\text{left-refinement-term } i \ j) \ (\text{left-refinement-term } i \ (\text{Suc } j)) \ (\cdot)$
1

definition $\text{right-refinement-factor-class} :: \text{nat} \Rightarrow \text{nat} \Rightarrow 'a \text{ set monoid-iso-class}$
where
 $\text{right-refinement-factor-class } j \ i \equiv$
 $\text{normal-factor-class } (\text{right-refinement-term } j \ i) \ (\text{right-refinement-term } j \ (\text{Suc } i))$
 $(\cdot) \ \mathbf{1}$

definition $\text{left-refinement-factor-classes} :: 'a \text{ set monoid-iso-class list}$
where
 $\text{left-refinement-factor-classes} \equiv$
 $\text{concat } (\text{List.map } (\lambda i. \text{List.map } (\text{left-refinement-factor-class } i) \ [0..<n]) \ [0..<m])$

definition $\text{right-refinement-factor-classes} :: 'a \text{ set monoid-iso-class list}$
where
 $\text{right-refinement-factor-classes} \equiv$
 $\text{concat } (\text{List.map } (\lambda j. \text{List.map } (\text{right-refinement-factor-class } j) \ [0..<m]) \ [0..<n])$

lemma $\text{length-left-refinement-factor-classes}$:
 $\text{length left-refinement-factor-classes} = m * n$
 $\langle \text{proof} \rangle$

lemma $\text{length-right-refinement-factor-classes}$:
 $\text{length right-refinement-factor-classes} = n * m$
 $\langle \text{proof} \rangle$

lemma $\text{refinement-factor-class-eq}$:
assumes $i: i < m$ **and** $j: j < n$
shows $\text{left-refinement-factor-class } i \ j = \text{right-refinement-factor-class } j \ i$
 $\langle \text{proof} \rangle$

The two row-major factor lists need not have the same order. Their multisets agree: transposing the refinement matrix sends the factor at position (i, j) to the isomorphic factor at (j, i) . This is the native, factor-class form of the Schreier refinement theorem.

theorem $\text{schreier-refinement}$:
 $\text{mset left-refinement-factor-classes} = \text{mset right-refinement-factor-classes}$
 $\langle \text{proof} \rangle$

end

16.4 Reduced Schreier refinements

A finite normal chain may repeat a term. Its corresponding quotient is the one-element group and contributes no genuine factor. The operation below removes

precisely the canonical trivial isomorphism class while retaining multiplicity and forgetting order.

definition *nontrivial-factor-multiset* ::
 'a monoid-iso-class list $\Rightarrow$ 'a monoid-iso-class multiset
where
 nontrivial-factor-multiset factors $\equiv$ *filter-mset* ($\lambda C. C \neq \text{trivial-monoid-iso-class}$)
 (*mset factors*)

lemma *nontrivial-factor-multiset-cong*:
assumes *mset factors* = *mset factors'*
shows *nontrivial-factor-multiset factors* = *nontrivial-factor-multiset factors'*
 <proof>

lemma *nontrivial-factor-multiset-append* [simp]:
nontrivial-factor-multiset (factors @ factors') =
nontrivial-factor-multiset factors + *nontrivial-factor-multiset factors'*
 <proof>

lemma *nontrivial-factor-multiset-Nil* [simp]:
nontrivial-factor-multiset [] = {#}
 <proof>

lemma *nontrivial-factor-multiset-singleton* [simp]:
nontrivial-factor-multiset [C] =
 (if *C* = *trivial-monoid-iso-class* then {#} else {#*C*#})
 <proof>

lemma *nontrivial-factor-multiset-concat-map-upt*:
*nontrivial-factor-multiset (concat (List.map f [0..*m*]))* =
 ($\sum i < m. \text{nontrivial-factor-multiset } (f\ i)$)
 <proof>

context *normal-series-pair*
begin

The trivial-class test has a concrete chain interpretation: it detects exactly an adjacent repetition. These equivalences are the bridge used by the later Jordan–Hölder argument when a refinement row lies inside a simple factor.

lemma *left-refinement-factor-class-eq-trivial-iff*:
assumes *i*: *i* < *m* and *j*: *j* < *n*
shows *left-refinement-factor-class i j* = *trivial-monoid-iso-class*
 $\longleftrightarrow$ *left-refinement-term i j* = *left-refinement-term i (Suc j)*
 <proof>

lemma *right-refinement-factor-class-eq-trivial-iff*:
assumes *i*: *i* < *m* and *j*: *j* < *n*
shows *right-refinement-factor-class j i* = *trivial-monoid-iso-class*
 $\longleftrightarrow$ *right-refinement-term j i* = *right-refinement-term j (Suc i)*
 <proof>

definition *left-reduced-refinement-factor-multiset* :: 'a set monoid-iso-class multiset
where *left-reduced-refinement-factor-multiset* $\equiv$ *nontrivial-factor-multiset left-refinement-factor-classes*

definition *right-reduced-refinement-factor-multiset* :: 'a set monoid-iso-class multiset
where *right-reduced-refinement-factor-multiset* $\equiv$ *nontrivial-factor-multiset right-refinement-factor-classes*

Removing repetition factors from equivalent Schreier refinements preserves equality. Unlike *mset left-refinement-factor-classes = mset right-refinement-factor-classes*, this form is ready to compare directly with the factor multiset of a composition series.

theorem *reduced-schreier-refinement*:
left-reduced-refinement-factor-multiset = right-reduced-refinement-factor-multiset
<proof>

end

end

17 Composition series, etc.

theory *Simple-Factor-Chain*
imports *Schreier-Refinement Normal-Series*
begin

17.1 Simple groups

A simple group is a non-trivial group whose only normal subgroups are the trivial subgroup and the whole group.

locale *Simple-Group* = *Group* *G* ($\cdot$) **1**
for *G* **and** *composition* (**infixl** $\langle \cdot \rangle$ 70) **and** *unit* ($\langle \mathbf{1} \rangle$) +
assumes *nontrivial*: $G \neq \{\mathbf{1}\}$
and *normal-subgroup-eq-trivial-or-top*:
 $\bigwedge K. \text{normal-subgroup } K \ G \ (\cdot) \ \mathbf{1} \implies K = \{\mathbf{1}\} \vee K = G$
begin

Simple groups have no proper non-trivial normal subgroups.

lemma *proper-normal-subgroup-is-trivial*:
assumes *K*: *normal-subgroup* *K* *G* ($\cdot$) **1** **and** *proper*: $K \neq G$
shows $K = \{\mathbf{1}\}$
<proof>

lemma *nontrivial-normal-subgroup-is-top*:
assumes *K*: *normal-subgroup* *K* *G* ($\cdot$) **1** **and** *K-nontrivial*: $K \neq \{\mathbf{1}\}$
shows $K = G$
<proof>

end

A simple factor group is the quotient associated with a normal subgroup whose factor group is simple. Naming this local construction keeps the quotient operations attached to the existing *normal-subgroup* interpretation.

```
locale simple-factor-group =
  N: normal-subgroup K H (·) 1 +
  Q: Simple-Group N.Factor-Group N.quotient-composition N.Class 1
  for K and H and composition (infixl <·> 70) and unit (<1>)
begin
```

```
lemma factor-group-is-simple:
  Simple-Group N.Factor-Group N.quotient-composition (N.Class 1)
  <proof>
```

```
lemma factor-group-nontrivial:
  N.Factor-Group ≠ {N.Class 1}
  <proof>
```

```
lemma proper:
  K ≠ H
  <proof>
```

end

A composition series is a finite normal series whose successive factor groups are simple. The indexed representation is inherited from *normal-series*.

```
locale composition-series = normal-series +
  assumes factor-simple:  $\bigwedge i. i < n \implies \text{simple-factor-group } (H\ i) (H\ (\text{Suc } i)) (\cdot) 1$ 
```

begin

```
lemma factor-simple-step:
  assumes i: i < n
  shows simple-factor-group (H i) (H (Suc i)) (·) 1
  <proof>
```

```
lemma series-factor-simple:
  assumes i: i < n
  shows Simple-Group (fst (series-factor i))
    (fst (snd (series-factor i))) (snd (snd (series-factor i)))
  <proof>
```

```
lemma series-factors-nth-simple:
  assumes i: i < n
  shows Simple-Group (fst (series-factors ! i))
    (fst (snd (series-factors ! i))) (snd (snd (series-factors ! i)))
  <proof>
```

lemma *composition-step-strict*:

assumes $i: i < n$
shows $H\ i \neq H\ (Suc\ i)$
 $\langle proof \rangle$

Every prefix of a composition series is again a composition series.

lemma *prefix-composition-series*:

assumes $k: k \leq n$
shows *composition-series* $(H\ k)\ (\cdot)\ \mathbf{1}\ H\ k$
 $\langle proof \rangle$

A composition series has positive length exactly when its group is nontrivial.

lemma *zero-length-iff-trivial*: $n = 0 \longleftrightarrow G = \{\mathbf{1}\}$
 $\langle proof \rangle$

end

17.2 Normal chains inside a simple factor

The subgroup correspondence detects the bottom and top subgroups of a quotient without choosing representatives. These two characterisations are the order-theoretic part of the correspondence needed below.

context *normal-subgroup-in-subgroup*
begin

lemma *Fract-H-K-eq-bottom-iff*:

$Fract\text{-}H\text{-}K = \{K.\text{Class}\ \mathbf{1}\} \longleftrightarrow H = K$
 $\langle proof \rangle$

lemma *Fract-H-K-eq-top-iff*:

$Fract\text{-}H\text{-}K = K.\text{Factor-Group} \longleftrightarrow H = G$
 $\langle proof \rangle$

end

An intermediate subgroup that is normal in the numerator of a simple factor is necessarily one of the endpoints. The proof passes to the quotient using the existing correspondence theorem and then reflects the two possible quotient subgroups back to the original carrier sets.

lemma *simple-factor-intermediate-normal*:

assumes *factor*: *simple-factor-group* $K\ H$ *composition unit*
and *contained*: $K \subseteq L$
and *normal*: *normal-subgroup* $L\ H$ *composition unit*
shows $L = K \vee L = H$
 $\langle proof \rangle$

definition *normal-chain-factor-classes* ::

$(nat \Rightarrow 'a\ set) \Rightarrow nat \Rightarrow$

$(\text{'a} \Rightarrow \text{'a} \Rightarrow \text{'a}) \Rightarrow \text{'a} \Rightarrow$
 $\text{'a set monoid-iso-class list}$

where

$\text{normal-chain-factor-classes } C \text{ } n \text{ composition unit} =$
 List.map
 $(\lambda i. \text{normal-factor-class } (C \text{ } i) (C \text{ } (\text{Suc } i)) \text{ composition unit})$
 $[0..<n]$

definition *reduced-normal-chain-factor-multiset* ::

$(\text{nat} \Rightarrow \text{'a set}) \Rightarrow \text{nat} \Rightarrow$
 $(\text{'a} \Rightarrow \text{'a} \Rightarrow \text{'a}) \Rightarrow \text{'a} \Rightarrow$
 $\text{'a set monoid-iso-class multiset}$

where

$\text{reduced-normal-chain-factor-multiset } C \text{ } n \text{ composition unit} =$
 $\text{nontrivial-factor-multiset}$
 $(\text{normal-chain-factor-classes } C \text{ } n \text{ composition unit})$

lemma *length-normal-chain-factor-classes*:

$\text{length } (\text{normal-chain-factor-classes } C \text{ } n \text{ composition unit}) = n$
 $\langle \text{proof} \rangle$

A finite normal chain between the endpoints of a simple factor has exactly one nontrivial factor. Repetitions may occur on either side of that strict step, but filtering the canonical trivial class leaves the original factor class with multiplicity one.

theorem *simple-factor-chain-reduction*:

assumes $\text{chain: normal-chain } G \text{ composition unit } C \text{ } n$
and $\text{factor: simple-factor-group } K \text{ } H \text{ composition unit}$
and $\text{bottom: } C \text{ } 0 = K$
and $\text{top: } C \text{ } n = H$
shows $\text{reduced-normal-chain-factor-multiset } C \text{ } n \text{ composition unit} =$
 $\{\# \text{normal-factor-class } K \text{ } H \text{ composition unit} \# \}$
 $\langle \text{proof} \rangle$

end

theory *Jordan-Hoelder-Uniqueness*

imports *Simple-Factor-Chain*

begin

18 The Jordan–Hölder theorem

Two composition series of the same set-based group form a normal-series pair, while simplicity of their factors supplies the additional rigidity needed to collapse each Schreier refinement row. Keeping these assumptions in one locale makes the final comparison symmetric and explicit.

locale *composition-series-pair* =

$\text{normal-series-pair } G \text{ } (\cdot) \text{ } \mathbf{1} \text{ } A \text{ } m \text{ } B \text{ } n +$
 $A\text{-comp: composition-series } G \text{ } (\cdot) \text{ } \mathbf{1} \text{ } A \text{ } m +$

```

B-comp: composition-series  $G (\cdot) \mathbf{1} B n$ 
for  $G$  and composition (infixl  $\langle \cdot \rangle$  70) and unit ( $\langle \mathbf{1} \rangle$ )
  and  $A :: \text{nat} \Rightarrow 'a \text{ set}$  and  $m$ 
  and  $B :: \text{nat} \Rightarrow 'a \text{ set}$  and  $n$ 
begin

```

Each refinement row spans one simple composition factor. The generic simple-factor chain theorem therefore leaves exactly that factor after deleting repetitions.

```

lemma left-refinement-row-reduction:
  assumes  $i: i < m$ 
  shows nontrivial-factor-multiset (List.map (left-refinement-factor-class  $i$ )  $[0..<n]$ )
=
  { $\#A.\text{series-factor-class } i \#$ }
<proof>

```

```

lemma right-refinement-row-reduction:
  assumes  $j: j < n$ 
  shows nontrivial-factor-multiset (List.map (right-refinement-factor-class  $j$ )  $[0..<m]$ )
=
  { $\#B.\text{series-factor-class } j \#$ }
<proof>

```

```

lemma left-reduced-refinement-eq-series-factor-multiset:
  left-reduced-refinement-factor-multiset = A.series-factor-multiset
<proof>

```

```

lemma right-reduced-refinement-eq-series-factor-multiset:
  right-reduced-refinement-factor-multiset = B.series-factor-multiset
<proof>

```

The factor-class multiset of a composition series is independent of the series. Equality is obtained by identifying each original series with its reduced Schreier refinement and applying the reduced refinement theorem.

```

theorem jordan-hoelder:
  A.series-factor-multiset = B.series-factor-multiset
<proof>

```

```

corollary jordan-hoelder-length: m = n
<proof>

```

```

end

```

```

end
theory Maximal-Normal-Subgroup
  imports Simple-Factor-Chain
begin

```

19 Maximal normal subgroups

A maximal normal subgroup is proper and admits no proper normal enlargement. Stating maximality with non-strict inclusion makes the locale convenient to consume in finite-choice and composition-series arguments.

```

locale maximal-normal-subgroup =
  N: normal-subgroup K G (·) 1
  for K and G and composition (infixl <·> 70) and unit (<1>) +
  assumes proper: K ≠ G
  and maximal:
    ∧ L. normal-subgroup L G (·) 1 ⇒ K ⊆ L ⇒
      L = K ∨ L = G
begin

lemma strict:
  K ⊂ G
  ⟨proof⟩

end

```

19.1 The simple-quotient characterisation

The quotient of a group by a proper normal subgroup is simple exactly when the subgroup is maximal normal. The forward correspondence uses the quotient endpoint characterisations developed for simple factors; the reverse direction is precisely normal-subgroup rigidity inside a simple factor.

```

theorem maximal-normal-subgroup-iff-simple-factor-group:
  maximal-normal-subgroup K G composition unit ⇔
    simple-factor-group K G composition unit
  ⟨proof⟩

```

19.2 Existence in finite groups

In a finite group, every proper normal subgroup lies below a maximal normal subgroup. Finiteness is used only to select a maximal member of the finite set of proper normal enlargements; the selected subgroup is therefore also suitable for induction on the carrier cardinality.

```

theorem finite-maximal-normal-subgroup-above:
  assumes G: Group G composition unit
  and finite-G: finite G
  and K-normal: normal-subgroup K G composition unit
  and K-proper: K ≠ G
  obtains M where
    maximal-normal-subgroup M G composition unit K ⊆ M
  ⟨proof⟩

```

corollary finite-nontrivial-group-has-maximal-normal-subgroup:


```

assumes  $G$ : Group  $G$  composition unit
and  $\text{finite-}G$ : finite  $G$  and  $\text{nontrivial}$ :  $G \neq \{\text{unit}\}$ 
obtains  $M$  where maximal-normal-subgroup  $M$   $G$  composition unit
<proof>

end
theory Finite-Composition-Series
imports Maximal-Normal-Subgroup
begin

```

20 Composition series of finite groups

A composition series for a normal subgroup can be extended by one simple factor. The old terms retain their indices and the new ambient group is placed at the successor of the old length. Keeping this construction separate makes the finite-cardinality induction below reusable.

```

definition extend-composition-series ::
  ( $\text{nat} \Rightarrow 'a \text{ set} \Rightarrow \text{nat} \Rightarrow 'a \text{ set} \Rightarrow \text{nat} \Rightarrow 'a \text{ set}$ )
where
  extend-composition-series  $H$   $n$   $G$   $i = (\text{if } i \leq n \text{ then } H\ i \text{ else } G)$ 

```

```

lemma extend-composition-series-old [simp]:
   $i \leq n \implies \text{extend-composition-series } H\ n\ G\ i = H\ i$ 
<proof>

```

```

lemma extend-composition-series-top [simp]:
   $\text{extend-composition-series } H\ n\ G\ (\text{Suc } n) = G$ 
<proof>

```

```

theorem composition-series-extend-simple-factor:
assumes  $\text{series}$ : composition-series  $K$  composition unit  $H\ n$ 
and  $\text{factor}$ : simple-factor-group  $K$   $G$  composition unit
shows composition-series  $G$  composition unit
  ( $\text{extend-composition-series } H\ n\ G$ ) ( $\text{Suc } n$ )
<proof>

```

20.1 Existence

Every finite group has a composition series. For a nontrivial group, choose a maximal normal subgroup. Its quotient is simple, while its proper carrier has smaller cardinality, so the induction hypothesis supplies the prefix.

```

theorem finite-group-has-composition-series-ex:
assumes  $G$ : Group  $G$  composition unit and  $\text{finite-}G$ : finite  $G$ 
shows  $\exists H\ n.$  composition-series  $G$  composition unit  $H\ n$ 
<proof>

```

```

corollary finite-group-has-composition-series:

```

```

assumes G: Group G composition unit and finite-G: finite G
obtains H n where composition-series G composition unit H n
  ⟨proof⟩

```

```

end
theory Composition-Factors-Solvable
  imports Finite-Composition-Series Jordan-Hoelder-Uniqueness Solvable-Transfer
begin

```

21 Solvability and composition factors

21.1 Auxiliary group-theoretic bridges

The kernel of the canonical quotient map is the normal subgroup itself. Although this is implicit in the quotient construction, an explicit lemma is useful whenever a theorem is phrased using the generic homomorphism kernel, as is the solvable-extension theorem.

```

lemma (in normal-subgroup) quotient-map-group-epimorphism:
  group-epimorphism Class G (·) unit
  Factor-Group quotient-composition (Class unit)
  ⟨proof⟩

```

```

lemma (in normal-subgroup) quotient-map-kernel:
  group-homomorphism.Ker Class G (Class unit) = K
  ⟨proof⟩

```

A simple group is solvable exactly when it is abelian. The nontrivial direction uses simplicity on the derived subgroup: if that subgroup were the whole carrier, every term of the derived series would remain the whole carrier, contradicting solvability.

```

theorem simple-group-solvable-iff-abelian:
  assumes simple: Simple-Group G composition unit
  shows Group.solvable G composition unit  $\longleftrightarrow$ 
    Abelian-Group G composition unit
  ⟨proof⟩

```

21.2 The factor criterion

```

context composition-series
begin

```

```

definition abelian-factors :: bool
where
  abelian-factors  $\longleftrightarrow$  ( $\forall i < n.$ 
    Abelian-Group (fst (series-factor i))
    (fst (snd (series-factor i))) (snd (snd (series-factor i))))

```

```

lemma abelian-factorsI:

```

assumes $\bigwedge i. i < n \implies \text{Abelian-Group } (\text{fst } (\text{series-factor } i))$
 $(\text{fst } (\text{snd } (\text{series-factor } i))) (\text{snd } (\text{snd } (\text{series-factor } i)))$
shows *abelian-factors*
 $\langle \text{proof} \rangle$

lemma *abelian-factorsD*:

assumes *abelian: abelian-factors* **and** *i: i < n*
shows *Abelian-Group* $(\text{fst } (\text{series-factor } i))$
 $(\text{fst } (\text{snd } (\text{series-factor } i))) (\text{snd } (\text{snd } (\text{series-factor } i)))$
 $\langle \text{proof} \rangle$

Solvability is equivalent to abelianness of every composition factor. The forward implication passes solvability to each subgroup and then to its quotient. The reverse implication reconstructs solvability one extension at a time along the series.

theorem *solvable-iff-abelian-factors*:

$G.\text{solvable} \longleftrightarrow \text{abelian-factors}$
 $\langle \text{proof} \rangle$

end

21.3 Independence and finite-group formulations

context *composition-series-pair*

begin

The Jordan–Hölder theorem identifies the complete factor multisets. For abelianness, the solvability criterion gives the corresponding invariance immediately, without choosing isomorphisms between matched factors.

corollary *abelian-factors-iff*:

$A.\text{comp.abelian-factors} \longleftrightarrow B.\text{comp.abelian-factors}$
 $\langle \text{proof} \rangle$

end

For finite groups the existential and universal formulations are both available because a composition series always exists. The universal form makes independence of the chosen series explicit.

theorem *finite-group-solvable-iff-has-composition-series-with-abelian-factors*:

assumes *G: Group G composition unit* **and** *finite-G: finite G*
shows $\text{Group.solvable } G \text{ composition unit} \longleftrightarrow$
 $(\exists H n. \text{composition-series } G \text{ composition unit } H n \wedge$
 $\text{composition-series.abelian-factors composition unit } H n)$
 $\langle \text{proof} \rangle$

theorem *finite-group-solvable-iff-all-composition-series-have-abelian-factors*:

assumes *G: Group G composition unit* **and** *finite-G: finite G*
shows $\text{Group.solvable } G \text{ composition unit} \longleftrightarrow$

$(\forall H\ n.\ \text{composition-series } G\ \text{composition unit } H\ n \longrightarrow$
 $\text{composition-series.abelian-factors composition unit } H\ n)$
 <proof>

Read the other way, the criterion says an abelian group has only abelian composition factors: an abelian group is solvable, and solvability is equivalent to all factors being abelian.

lemma *abelian-group-composition-factors-abelian:*
assumes *ab: Abelian-Group G comp e*
and series: *composition-series G comp e H m*
shows *composition-series.abelian-factors comp e H m*
 <proof>

end

22 The Symmetric and Alternating Groups

theory *Symmetric-Group*
imports *Group-Theory HOL-Combinatorics.Permutations HOL-Combinatorics.Cycles*
begin

We realise the concrete symmetric group S_n as the group of permutations of $\{0..<n\}$ under function composition, reusing the permutation and sign machinery of *HOL-Combinatorics.Permutations*.

Design note: the development's own $\lambda S.\ \text{Monoid.Units } (S \rightarrow_E S)$ (*compose S*) (*identity S*) consists of *extensional* maps (undefined outside the carrier), which does not interoperate directly with the library predicate *p permutes S* (identity outside the carrier). Building S_n natively on $\{p.\ p\ \text{permutes } S\}$ avoids that impedance mismatch and lets us inherit *sign* and its homomorphism property for free.

22.1 The symmetric group as a Group

definition $S_n :: \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{nat})\ \text{set}$
where $S_n\ n = \{p.\ p\ \text{permutes } \{0..<n\}\}$

lemma *S_n-iff:* $p \in S_n\ n \longleftrightarrow p\ \text{permutes } \{0..<n\}$
 <proof>

lemma *S_n-permutation:* $p \in S_n\ n \implies \text{permutation } p$
 <proof>

$S_n\ n$ is a group under composition.

theorem *Group-S_n:* $\text{Group } (S_n\ n)\ (\circ)\ \text{id}$
 <proof>

22.2 Relation to the development's abstract symmetric group

The development's own symmetric group $\text{Monoid.Units } (S \rightarrow_E S) \text{ (compose } S) \text{ (identity } S)$ consists of the *extensional* bijections of S under *compose* $S/\text{identity } S$. Our $\text{Sn } n$ uses library (*permutes*) maps (identity outside). The two are reconciled by *restrict*: restricting a permutation to S yields an element of $\text{Monoid.Units } (S \rightarrow_E S) \text{ (compose } S) \text{ (identity } S)$, and this correspondence is bijective. This makes the concrete $\text{Sn } n$ available to the development's *Group-Action* machinery.

lemma *restrict-permutes-in-Sym*:

assumes $p \text{ permutes } S$
shows $\text{restrict } p \text{ } S \in \text{transformations.Sym } S$
 $\langle \text{proof} \rangle$

Conversely, an element of $\text{Monoid.Units } (S \rightarrow_E S) \text{ (compose } S) \text{ (identity } S)$ permutes S once it is extended by the identity outside S .

lemma *Sym-extend-permutes*:

assumes $\alpha \in \text{transformations.Sym } S$
shows $(\lambda x. \text{ if } x \in S \text{ then } \alpha \ x \text{ else } x) \text{ permutes } S$
 $\langle \text{proof} \rangle$

22.3 The sign homomorphism and the alternating group

The two-element group of signs.

lemma *Group-signs*: $\text{Group } \{1, -1::\text{int}\} (*) 1$
 $\langle \text{proof} \rangle$

lemma *sign-range*: $p \in \text{Sn } n \implies \text{sign } p \in \{1, -1::\text{int}\}$
 $\langle \text{proof} \rangle$

sign is a group homomorphism $S_n \rightarrow \{1, -1\}$. The multiplicativity is inherited directly from *sign-compose*.

lemma *sign-hom*: $\text{group-homomorphism } (\text{restrict sign } (\text{Sn } n)) (\text{Sn } n) (\circ) \text{ id } \{1, -1::\text{int}\} (*) 1$
 $\langle \text{proof} \rangle$

The alternating group is the kernel of the sign homomorphism.

definition $\text{An } n :: \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{nat}) \text{ set}$
where $\text{An } n = \text{group-homomorphism.Ker } (\text{restrict sign } (\text{Sn } n)) (\text{Sn } n) (1::\text{int})$

context

fixes $n::\text{nat}$

begin

interpretation H : $\text{group-homomorphism } \text{restrict sign } (\text{Sn } n) \text{ Sn } n (\circ) \text{ id } \{1, -1::\text{int}\} (*) 1$

<proof>

lemma *An-normal: normal-subgroup* ($An\ n$) ($Sn\ n$) ($\circ$) *id*
<proof>

lemma *An-iff*: $p \in An\ n \longleftrightarrow p \in Sn\ n \wedge sign\ p = 1$
<proof>

end

lemma *An-subset-Sn*: $An\ n \subseteq Sn\ n$
<proof>

22.4 Three-cycles

A three-cycle on distinct points of $\{0..<n\}$ is a permutation, and it is even, hence lies in the alternating group.

lemma *three-cycle-permutes*:
 assumes *distinct* $[a,b,c]$ $a < n\ b < n\ c < n$
 shows *cycle-of-list* $[a,b,c]$ *permutes* $\{0..<(n::nat)\}$
<proof>

lemma *three-cycle-in-Sn*:
 assumes *distinct* $[a,b,c]$ $a < n\ b < n\ c < n$
 shows *cycle-of-list* $[a,b,c] \in Sn\ n$
<proof>

lemma *three-cycle-sign*:
 assumes *distinct* $[a,b,c]$
 shows *sign* (*cycle-of-list* $[a,b,c]$) = 1
<proof>

lemma *three-cycle-in-An*:
 assumes *distinct* $[a,b,c]$ $a < n\ b < n\ c < n$
 shows *cycle-of-list* $[a,b,c] \in An\ n$
<proof>

end

23 Insolubility of the Symmetric Group

theory *Sym-Not-Solvable*
 imports *Symmetric-Group Solvable-Transfer*
begin

For $n \geq 5$ the symmetric group $Sn\ n$ is not solvable. The argument is the classical one: every three-cycle is a commutator of two three-cycles (using two “spare” points, which exist when $n \geq 5$), so the three-cycles survive in every

term of the derived series, which therefore never reaches the trivial subgroup.

23.1 Three-cycles as commutators

The inverse of a three-cycle is the reversed three-cycle.

lemma *rev-inv-3cycle*:
assumes *distinct* [a,b,(c::nat)]
shows *inv* (cycle-of-list [a,b,c]) = cycle-of-list [c,b,a]
 <proof>

Inside S_n the group inverse coincides with the functional inverse.

lemma *Sn-inverse*:
assumes $p \in S_n$
shows *Monoid.inverse* (S_n) ($\circ$) *id* $p = \text{inv } p$
 <proof>

The defining pointwise identity: a three-cycle factors as a product of four three-cycles (which is, up to inverses, a commutator). Verified by a six-way case split on the argument.

lemma *three-cycle-commutator-core*:
assumes *dist*: *distinct* [a,b,c,d,(e::nat)]
shows cycle-of-list [a,b,c]
 = cycle-of-list [a,b,d] $\circ$ cycle-of-list [c,e,a] $\circ$ cycle-of-list [d,b,a] $\circ$ cycle-of-list
 [a,e,c]
 <proof>

Hence a three-cycle is a commutator of two three-cycles in S_n .

theorem *three-cycle-commutator*:
assumes *dist*: *distinct* [a,b,c,d,(e::nat)] **and** *lt*: $a < n$ $b < n$ $c < n$ $d < n$ $e < n$
shows cycle-of-list [a,b,c]
 = *Group.commutator-elt* (S_n) ($\circ$) *id* (cycle-of-list [a,b,d]) (cycle-of-list
 [c,e,a])
 <proof>

23.2 Persistence of three-cycles in the derived series

A set of cardinality at least two contains two distinct elements. (A direct consequence of the library lemma *card-le-Suc-iff*.)

lemma *card-ge2-two-elems*:
assumes $2 \leq \text{card } R$
shows $\exists d \ e. d \in R \wedge e \in R \wedge d \neq e$
 <proof>

For $n \geq 5$ there are two further points distinct from any three.

lemma *two-spare-points*:
assumes *n5*: $5 \leq n$ **and** *dist*: *distinct* [a,b,(c::nat)] **and** *lt*: $a < n$ $b < n$ $c < n$

shows $\exists d e. \text{distinct } [a,b,c,d,e] \wedge d < n \wedge e < n$
 $\langle \text{proof} \rangle$

If a subgroup of S_n contains all three-cycles, so does its commutator subgroup.

lemma *three-cycle-in-commutator-subgroup*:

assumes $n5: 5 \leq n$
and $H: \text{Subgroup } H (S_n n) (\circ) \text{ id}$
and $Hcyc: \bigwedge x y z. \text{distinct } [x,y,z] \implies x < n \implies y < n \implies z < n \implies \text{cycle-of-list } [x,y,z] \in H$
and $dist: \text{distinct } [a,b,c]$ **and** $lt: a < n \wedge b < n \wedge c < n$
shows $\text{cycle-of-list } [a,b,c] \in \text{Group.commutator-subgroup } (S_n n) (\circ) \text{ id } H H$
 $\langle \text{proof} \rangle$

Every three-cycle survives in every term of the derived series of S_n .

lemma *three-cycles-in-derivedSeries*:

assumes $n5: 5 \leq n$ **and** $dist: \text{distinct } [a,b,c]$ **and** $lt: a < n \wedge b < n \wedge c < n$
shows $\text{cycle-of-list } [a,b,c] \in \text{Group.derivedSeries } (S_n n) (\circ) \text{ id } m$
 $\langle \text{proof} \rangle$

23.3 The main theorem

lemma *three-cycle-012-neq-id*: $\text{cycle-of-list } [0,1,2] \neq (\text{id} :: \text{nat} \Rightarrow \text{nat})$
 $\langle \text{proof} \rangle$

theorem *S_n -not-solvable*:

assumes $n5: 5 \leq n$
shows $\neg \text{Group.solvable } (S_n n) (\circ) \text{ id}$
 $\langle \text{proof} \rangle$

23.4 The alternating group

The same argument runs inside the alternating group: every three-cycle is even, so it lies in A_n , and the commutator-of-three-cycles identity stays inside A_n . The key reuse is *Group.commutator-elt-subgroup-eq*: the commutator computed in the subgroup A_n agrees with the one in S_n .

lemma *A_n -is-subgroup-of- S_n* : $\text{Subgroup } (A_n n) (S_n n) (\circ) \text{ id}$
 $\langle \text{proof} \rangle$

lemma *Group- A_n* : $\text{Group } (A_n n) (\circ) \text{ id}$
 $\langle \text{proof} \rangle$

lemma *three-cycle-in- A_n -commutator-subgroup*:

assumes $n5: 5 \leq n$
and $H: \text{Subgroup } H (A_n n) (\circ) \text{ id}$
and $Hcyc: \bigwedge x y z. \text{distinct } [x,y,z] \implies x < n \implies y < n \implies z < n \implies \text{cycle-of-list } [x,y,z] \in H$
and $dist: \text{distinct } [a,b,c]$ **and** $lt: a < n \wedge b < n \wedge c < n$
shows $\text{cycle-of-list } [a,b,c] \in \text{Group.commutator-subgroup } (A_n n) (\circ) \text{ id } H H$

<proof>

lemma *three-cycles-in-An-derivedSeries:*

assumes *n5*: $5 \leq n$ **and** *dist*: *distinct* $[a, b, c]$ **and** *lt*: $a < n \ b < n \ c < n$

shows *cycle-of-list* $[a, b, c] \in \text{Group.derivedSeries } (An \ n) \ (\circ) \ id \ m$

<proof>

theorem *An-not-solvable:*

assumes *n5*: $5 \leq n$

shows $\neg \text{Group.solvable } (An \ n) \ (\circ) \ id$

<proof>

end

24 Generating the Symmetric Group

theory *Symmetric-Generation*

imports *Symmetric-Group Sylow-Theorems*

begin

Sn n is finite: it is a set of permutations of a finite set. Stated here at the top level — it was previously inside the anonymous context below that fixes a *p*-cycle *g*, so the exported fact carried that context's four irrelevant assumptions.

lemma *finite-Sn*: *finite* $(Sn \ n)$

<proof>

The symmetric group *Sn n* is generated by a single transposition together with a full *n*-cycle, and (for prime *p*) by *any* transposition together with *any* *p*-cycle. These generation results are the combinatorial ingredient that, combined with *Sn-not-solvable*, shows that the Galois group of a quintic with exactly two non-real roots is the whole of *Sn 5*.

24.1 A general lemma on generated subgroups

Powers of an element of a generated subgroup stay inside it.

lemma (**in** *Group*) *generate-funpow*:

assumes $g \in \text{generate } S$

shows *power* $g \ k \in \text{generate } S$

<proof>

24.2 Conjugation is an automorphism of the generated subgroup

Conjugation by a group element *g* fixes the unit and commutes with generation: the image of *generate S* under *conjugation g* is the subgroup generated by the image of *S*. This lets a generation result be transported along conjugacy, which is what upgrades the standard-cycle generation theorem to an *arbitrary* cycle.

lemma (in Group) conjugation-one: $g \in G \implies \text{conjugation } g \ 1 = 1$
 ⟨proof⟩

lemma (in Group) conjugation-inv-cancel:
assumes $g: g \in G$ **and** $x: x \in G$
shows $\text{conjugation } (\text{inverse } g) (\text{conjugation } g \ x) = x$
 ⟨proof⟩

lemma (in Group) conjugation-generate-sub:
assumes $g: g \in G$
shows $\text{conjugation } g \text{ ' generate } S \subseteq \text{generate } (\text{conjugation } g \text{ ' } (S \cap G))$
 ⟨proof⟩

lemma (in Group) conjugation-generate-eq:
assumes $g: g \in G$
shows $\text{conjugation } g \text{ ' generate } S = \text{generate } (\text{conjugation } g \text{ ' } (S \cap G))$
 ⟨proof⟩

24.3 Transpositions generate the symmetric group

The set of transpositions of points of $\{0..<n\}$.

definition *transpositions* :: $\text{nat} \Rightarrow (\text{nat} \Rightarrow \text{nat}) \text{ set}$
where *transpositions* $n = \{ \text{Transposition.transpose } a \ b \mid a \ b. \ a \in \{0..<n\} \wedge b \in \{0..<n\} \}$

lemma *transpositions-subset-Sn*: $\text{transpositions } n \subseteq S_n \ n$
 ⟨proof⟩

24.4 The standard n -cycle

The cyclic shift sending i to $\text{Suc } i \text{ mod } n$ on $\{0..<n\}$, fixing everything outside. We work with this explicit form rather than *cycle-of-list* because its powers have the transparent closed form $(i + k) \text{ mod } n$.

definition *ncycle* :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat}$
where *ncycle* $n \ i = (\text{if } i < n \text{ then } \text{Suc } i \text{ mod } n \text{ else } i)$

lemma *ncycle-lt*: $i < n \implies \text{ncycle } n \ i < n$ ⟨proof⟩

lemma *ncycle-inv-right*: $i < n \implies \text{ncycle } n \ ((i + (n-1)) \text{ mod } n) = i$
 ⟨proof⟩

lemma *ncycle-inv-left*: $i < n \implies (\text{ncycle } n \ i + (n-1)) \text{ mod } n = i$
 ⟨proof⟩

lemma *ncycle-permutes*: $\text{ncycle } n \text{ permutes } \{0..<n\}$
 ⟨proof⟩

lemma *ncycle-in-Sn*: $\text{ncycle } n \in S_n \ n$ ⟨proof⟩

lemma *ncycle-bij*: *bij (ncycle n)*
 ⟨proof⟩

The closed form for the iterates of the n -cycle.

lemma *ncycle-funpow*: $i < n \implies (\text{ncycle } n \text{ } \sim^k) i = (i + k) \bmod n$
 ⟨proof⟩

context
fixes $n::\text{nat}$
begin

interpretation S : *Group $S_n n$ ($\circ$) id*
 ⟨proof⟩

Every permutation of $\{0..<n\}$ is a product of transpositions, so the transpositions generate $S_n n$. This follows directly from the tailored induction rule *permutes-induct*.

lemma *S_n -generated-by-transpositions*:
Group.generate ($S_n n$) ($\circ$) id (transpositions n) = $S_n n$
 ⟨proof⟩

24.5 Conjugating a transposition

Conjugating a transposition by a bijection transposes the images of the two points.

lemma *transpose-conjug*:
assumes *bij q*
shows $q \circ \text{Transposition.transpose } a \ b \circ \text{inv } q = \text{Transposition.transpose } (q \ a) \ (q \ b)$
 ⟨proof⟩

24.6 The symmetric group is generated by a transposition and a cycle

The group power in $S_n n$ (composition) is just function iteration.

lemma *S_n -power-funpow*: *Monoid.power ($\circ$) id ($g :: \text{nat} \Rightarrow \text{nat}$) $k = g \text{ } \sim^k$*
 ⟨proof⟩

In $S_n n$ the group inverse is the functional inverse.

lemma *S_n -inverse-eq*:
assumes $p \in S_n n$
shows *Monoid.inverse ($S_n n$) ($\circ$) id $p = \text{inv } p$*
 ⟨proof⟩

lemma *t01-in- S_n* : $1 < n \implies \text{Transposition.transpose } 0 \ 1 \in S_n n$
 ⟨proof⟩

Conjugating $\text{Transposition.transpose } 0\ 1$ by the k -th power of the n -cycle yields the adjacent transposition $\text{Transposition.transpose } (k \bmod n)\ (Suc\ k \bmod n)$.

lemma *adjacent-transpose-conjug*:

assumes $k < n\ 1 < n$

shows $(\text{ncycle } n \rightsquigarrow k) \circ \text{Transposition.transpose } 0\ 1 \circ \text{inv } (\text{ncycle } n \rightsquigarrow k)$
 $= \text{Transposition.transpose } (k \bmod n)\ (Suc\ k \bmod n)$

<proof>

Hence each adjacent transposition lies in the subgroup generated by $\text{Transposition.transpose } 0\ 1$ and the n -cycle.

lemma *adjacent-in-generate*:

assumes $k < n\ 1 < n$

defines $S \equiv \{\text{Transposition.transpose } 0\ 1, \text{ncycle } n\}$

shows $\text{Transposition.transpose } (k \bmod n)\ (Suc\ k \bmod n) \in \text{Group.generate } (Sn\ n)$

(o) id S

<proof>

Building from adjacent transpositions, every transposition $\text{transpose } i\ j$ with $i < j < n$ lies in the subgroup.

lemma *transpose-extend*:

assumes $a \neq b\ b \neq c\ a \neq c$

shows $\text{Transposition.transpose } a\ c = \text{Transposition.transpose } a\ b \circ \text{Transposition.transpose } b\ c \circ \text{Transposition.transpose } a\ b$

<proof>

lemma *all-transpose-in-generate-aux*:

assumes $1 < n$

defines $S \equiv \{\text{Transposition.transpose } 0\ 1, \text{ncycle } n\}$

shows $\forall i. i + Suc\ d < n \longrightarrow \text{Transposition.transpose } i\ (i + Suc\ d) \in \text{Group.generate } (Sn\ n)$

(o) id S

<proof>

lemma *transpose-in-generate*:

assumes $a < n\ b < n\ 1 < n$

shows $\text{Transposition.transpose } a\ b \in \text{Group.generate } (Sn\ n) \circ \text{id } \{\text{Transposition.transpose } 0\ 1, \text{ncycle } n\}$

<proof>

The canonical generation theorem: a transposition of two adjacent points together with the standard n -cycle generate the whole symmetric group.

theorem *Sn-generated-canonical*:

assumes $1 < n$

shows $\text{Group.generate } (Sn\ n) \circ \text{id } \{\text{Transposition.transpose } 0\ 1, \text{ncycle } n\} = Sn\ n$

<proof>

24.7 A transposition and a p -cycle generate S_p for prime p

For prime p , the standard p -cycle together with *any* transposition generates S_n . The primality is essential: it guarantees that the conjugates of the transposition by powers of the cycle have supports forming a single orbit covering all points, from which every transposition can be recovered.

Conjugating *Transposition.transpose a b* by the k -th power of the n -cycle shifts both points by k (modulo n); the result stays in the subgroup generated by the cycle and the transposition.

lemma *ncycle-conj-transpose-in-generate*:

assumes $a < n$ $b < n$ $1 < n$
defines $S \equiv \{\text{ncycle } n, \text{Transposition.transpose } a \ b\}$
shows $\text{Transposition.transpose } ((a + k) \bmod n) ((b + k) \bmod n) \in \text{Group.generate } (S \ n \ n) \ (\circ) \ \text{id } S$
 $\langle \text{proof} \rangle$

If a subgroup contains two transpositions sharing a point, it contains their “sum”.

lemma *transpose-chain*:

assumes H : Subgroup H $(S \ n \ n) \ (\circ) \ \text{id}$
and $\text{Transposition.transpose } x \ y \in H$
and $\text{Transposition.transpose } y \ z \in H$
and $x \neq z$ $y \neq z$
shows $\text{Transposition.transpose } x \ z \in H$
 $\langle \text{proof} \rangle$

end

context

fixes $p::\text{nat}$
assumes p : prime p

begin

interpretation G : Group $S \ n \ p \ (\circ) \ \text{id}$
 $\langle \text{proof} \rangle$

lemma *p-gt1*: $1 < p$
 $\langle \text{proof} \rangle$

Number-theoretic facts about the stride d (the gap of the transposition).

lemma *coprime-prime-stride*:

assumes $0 < d$ $d < p$
shows coprime $p \ d$
 $\langle \text{proof} \rangle$

lemma *kd-mod-zero-iff*:

assumes $0 < d \wedge d < p$
shows $(k * d) \bmod p = 0 \iff p \text{ dvd } k$
 $\langle \text{proof} \rangle$

lemma *stride-endpoints-distinct*:

assumes $0 < d \wedge d < p$
shows $(k * d) \bmod p \neq (\text{Suc } k * d) \bmod p$
 $\langle \text{proof} \rangle$

Each “stride edge” $\text{transpose } (k * d \bmod p) (\text{Suc } k * d \bmod p)$ lies in the subgroup generated by the cycle and $\text{Transposition.transpose } 0 \ d$.

lemma *stride-edge*:

assumes $d1: 0 < d$ **and** $d2: d < p$
defines $S \equiv \{\text{ncycle } p, \text{Transposition.transpose } 0 \ d\}$
shows $\text{Transposition.transpose } ((k * d) \bmod p) ((\text{Suc } k * d) \bmod p) \in \text{Group.generate } (S \text{ n } p) (\circ) \text{ id } S$
 $\langle \text{proof} \rangle$

Chaining the stride edges, every $\text{transpose } 0 \ (k * d \bmod p)$ (for k not a multiple of p) lies in the subgroup.

lemma *stride-walk*:

fixes $d :: \text{nat}$
assumes $d1: 0 < d$ **and** $d2: d < p$
defines $S \equiv \{\text{ncycle } p, \text{Transposition.transpose } 0 \ d\}$
shows $\neg p \text{ dvd } k \implies \text{Transposition.transpose } 0 \ ((k * d) \bmod p) \in \text{Group.generate } (S \text{ n } p) (\circ) \text{ id } S$
 $\langle \text{proof} \rangle$

lemma *pymod*: $(p * y + 1) \bmod p = (1 :: \text{nat}) \bmod p$
 $\langle \text{proof} \rangle$

Since d is coprime to the prime p , some multiple of d is congruent to 1 , so $\text{Transposition.transpose } 0 \ 1$ is reached.

lemma *stride-reaches-one*:

fixes $d :: \text{nat}$
assumes $d1: 0 < d$ **and** $d2: d < p$
shows $\exists k. (k * d) \bmod p = 1 \wedge \neg p \text{ dvd } k$
 $\langle \text{proof} \rangle$

lemma *t01-in-stride-generate*:

assumes $0 < d \wedge d < p$
shows $\text{Transposition.transpose } 0 \ 1 \in \text{Group.generate } (S \text{ n } p) (\circ) \text{ id } \{\text{ncycle } p, \text{Transposition.transpose } 0 \ d\}$
 $\langle \text{proof} \rangle$

The cycle and a stride- d transposition generate $S \text{ n } p$.

theorem *Sn-generated-cycle-transpose-0d*:

fixes $d :: \text{nat}$

assumes $d1: 0 < d$ **and** $d2: d < p$
shows $\text{Group.generate } (Sn\ p) (\circ) id \{\text{ncycle } p, \text{Transposition.transpose } 0\ d\} = Sn\ p$
 $\langle proof \rangle$

lemma *sum-pa-bounds*:

fixes $a\ b :: nat$
assumes $a < p$ $b < p$ $a \neq b$
shows $0 < (b + (p - a)) \bmod p$
 $\langle proof \rangle$

The main result of this section: for prime p , the standard p -cycle and *any* transposition generate $Sn\ p$.

theorem *Sn-generated-ncycle-transpose*:

fixes $a\ b :: nat$
assumes $a: a < p$ **and** $b: b < p$ **and** $ab: a \neq b$
shows $\text{Group.generate } (Sn\ p) (\circ) id \{\text{ncycle } p, \text{Transposition.transpose } a\ b\} = Sn\ p$
 $\langle proof \rangle$

24.8 An arbitrary cycle and transposition generate S_p

In $Sn\ n$ the group conjugation specialises to the familiar $g \circ x \circ inv\ g$.

lemma *Sn-conjugation-eq*:

assumes $g: g \in Sn\ n$ **and** $x: x \in Sn\ n$
shows $\text{Group.conjugation } (Sn\ n) (\circ) id\ g\ x = g \circ x \circ inv\ g$
 $\langle proof \rangle$

Transporting *Sn-generated-ncycle-transpose* along conjugacy: for prime p , *any* conjugate $q \circ \text{ncycle } p \circ inv\ q$ of the standard p -cycle, together with the conjugate $q \circ \text{Transposition.transpose } a\ b \circ inv\ q$ of a transposition, generates $Sn\ p$.

lemma *Sn-generated-conj-cycle-transpose*:

fixes $q :: nat \Rightarrow nat$ **and** $a\ b :: nat$
assumes $q: q \in Sn\ p$ **and** $a: a < p$ **and** $b: b < p$ **and** $ab: a \neq b$
shows $\text{Group.generate } (Sn\ p) (\circ) id$
 $\{q \circ \text{ncycle } p \circ inv\ q, q \circ \text{Transposition.transpose } a\ b \circ inv\ q\} = Sn\ p$
 $\langle proof \rangle$

24.9 An order- p element of S_p is a p -cycle

The combinatorial counterpart of the previous result: every element of $Sn\ p$ of order p (for p prime) is conjugate to the standard cycle, hence is itself a p -cycle. Together with *Sn-generated-conj-cycle-transpose* this lets Cauchy's theorem (which only yields *some* order- p element) feed the generation argument for the quintic.

Iterating g is periodic with period dividing n once $g^n = id$.

lemma *funpow-mod-period*:

assumes $(g \text{ } \smile \text{ } n) = id$
shows $(g \text{ } \smile \text{ } (k \bmod n)) = (g \text{ } \smile \text{ } k)$
 $\langle proof \rangle$

If g has order p (prime) and moves x , then no smaller positive power fixes x :
the only powers fixing x are the multiples of p .

lemma *cycle-period-prime*:
assumes p : prime p **and** gp : $(g \text{ } \smile \text{ } p) = id$ **and** gx : $g\ x \neq x$
and d : $(g \text{ } \smile \text{ } d)\ x = x$
shows $d \bmod p = 0$
 $\langle proof \rangle$

The conjugator: the map sending i to $g^i\ x$ on $\{0..<p\}$ and fixing the rest.

definition $qmap :: (nat \Rightarrow nat) \Rightarrow nat \Rightarrow nat \Rightarrow nat$
where $qmap\ g\ x\ i = (if\ i < p\ then\ (g \text{ } \smile \text{ } i)\ x\ else\ i)$

context
fixes $g :: nat \Rightarrow nat$ **and** $x :: nat$
assumes gSn : $g \in Sn\ p$ **and** gp : $(g \text{ } \smile \text{ } p) = id$
and gx : $g\ x \neq x$ **and** xp : $x < p$
begin

lemma *g-funpow-lt*: $i < p \implies (g \text{ } \smile \text{ } i)\ x < p$
 $\langle proof \rangle$

lemma *qmap-inj-aux*:
assumes ij : $i < p\ j < p\ i \leq j$ **and** eq : $(g \text{ } \smile \text{ } i)\ x = (g \text{ } \smile \text{ } j)\ x$
shows $i = j$
 $\langle proof \rangle$

lemma *qmap-inj-on*: $inj\ on\ (qmap\ g\ x)\ \{0..<p\}$
 $\langle proof \rangle$

lemma *qmap-image*: $qmap\ g\ x\ ` \{0..<p\} = \{0..<p\}$
 $\langle proof \rangle$

lemma *qmap-permutes*: $qmap\ g\ x\ permutes\ \{0..<p\}$
 $\langle proof \rangle$

lemma *qmap-in-Sn*: $qmap\ g\ x \in Sn\ p$
 $\langle proof \rangle$

lemma *g-qmap-eq*:
assumes $i < p$
shows $g\ (qmap\ g\ x\ i) = qmap\ g\ x\ (ncycle\ p\ i)$
 $\langle proof \rangle$

lemma *g-comp-qmap*: $g \circ qmap\ g\ x = qmap\ g\ x \circ ncycle\ p$
 $\langle proof \rangle$


```

lemma g-conj-ncycle:  $g = \text{qmap } g \ x \circ \text{ncycle } p \circ \text{inv } (\text{qmap } g \ x)$ 
  <proof>

```

```

end

```

```

theorem order-p-elt-conj-ncycle:
  fixes  $g :: \text{nat} \Rightarrow \text{nat}$ 
  assumes  $gSn: g \in Sn \ p$  and  $gp: (g \ \sim \ p) = id$  and  $gne: g \neq id$ 
  shows  $\exists q \in Sn \ p. g = q \circ \text{ncycle } p \circ \text{inv } q$ 
  <proof>

```

The same conclusion phrased through the group's *Monoid.element-order*: an element of $Sn \ p$ of order p is a conjugate of the standard cycle.

```

theorem order-p-elt-conj-ncycle-order:
  fixes  $g :: \text{nat} \Rightarrow \text{nat}$ 
  assumes  $gSn: g \in Sn \ p$ 
  and  $ord: \text{Monoid.element-order } (\circ) \ id \ g = p$ 
  shows  $\exists q \in Sn \ p. g = q \circ \text{ncycle } p \circ \text{inv } q$ 
  <proof>

```

```

end

```

```

end

```

25 Composition factors of the symmetric and alternating groups

```

theory Sn-Composition-Factors
  imports Composition-Factors-Solvable Sym-Not-Solvable Symmetric-Generation
begin

```

These are *consistency* checks, not new mathematics. The composition-series material of *Composition-Factors-Solvable* was contributed as a large block and merged on structural evidence (it builds, contains no *sorry*, and cites the existing solvability lemmas). The checks below test its *content* against results proved independently elsewhere in this development, on the principle that a vacuous or too-weak definition would let a false statement through.

The sharpest available cross-check is *Sn-not-solvable*: this development already proves, by an argument that knows nothing of composition series, that S_n is not solvable for $n \geq 5$. The new criterion says solvability is equivalent to all composition factors being abelian. The two must agree, and forcing them to agree pins down real content: if *composition-series.abelian-factors* were vacuously true — say because *series-factor* returned something degenerate — then S_5 would come out solvable and contradict a theorem already in the library.

25.1 A finite group has a composition series, and S_5 in particular

First that the existence theorem applies at all to a concrete group of interest.

lemma *Sn-has-composition-series:*

$\exists H m. \text{composition-series } (Sn\ n) \ (\circ) \ id\ H\ m$
 $\langle proof \rangle$

25.2 The criterion is consistent with S_n being unsolvable

The main cross-check. Every composition series of S_n ($n \geq 5$) has a factor that is *not* abelian. This is not proved by exhibiting one — it is forced, by combining the new criterion with the independently proved unsolvability of S_n . It therefore fails if the new criterion is too weak in either direction.

theorem *Sn-composition-factors-not-all-abelian:*

assumes $n5: 5 \leq n$ **and** *series: composition-series* $(Sn\ n) \ (\circ) \ id\ H\ m$
shows $\neg \text{composition-series.abelian-factors } (\circ) \ id\ H\ m$
 $\langle proof \rangle$

The same for the alternating groups, via *An-not-solvable*.

theorem *An-composition-factors-not-all-abelian:*

assumes $n5: 5 \leq n$ **and** *series: composition-series* $(An\ n) \ (\circ) \ id\ H\ m$
shows $\neg \text{composition-series.abelian-factors } (\circ) \ id\ H\ m$
 $\langle proof \rangle$

And the existence and criterion theorems combine: some composition series of S_5 exists, and every one of them has a non-abelian factor. If *composition-series* were unsatisfiable these two would not sit together — the first would be vacuous.

corollary *S5-has-series-with-a-nonabelian-factor:*

$\exists H m. \text{composition-series } (Sn\ 5) \ (\circ) \ id\ H\ m$
 $\wedge \neg \text{composition-series.abelian-factors } (\circ) \ id\ H\ m$
 $\langle proof \rangle$

end

26 Subfields and Field Homomorphisms (Type-Class Setting)

theory *Subfield*

imports *HOL-Computational-Algebra.Polynomial*
begin

We develop field-extension theory generically over an ambient field $'a$, taking a *subfield* to be a subset closed under the field operations..)

26.1 The subfield generated by a set

inductive-set *generate-field* :: 'a :: field set $\Rightarrow$ 'a set **for** *K* :: 'a set **where**
 generate-field-base [intro]: $x \in K \Rightarrow x \in \text{generate-field } K$
 | *generate-field-0* [intro]: $0 \in \text{generate-field } K$
 | *generate-field-1* [intro]: $1 \in \text{generate-field } K$
 | *generate-field-minus* [intro]: $x \in \text{generate-field } K \Rightarrow -x \in \text{generate-field } K$
 | *generate-field-add* [intro]: $x \in \text{generate-field } K \Rightarrow y \in \text{generate-field } K \Rightarrow x + y \in \text{generate-field } K$
 | *generate-field-mult* [intro]: $x \in \text{generate-field } K \Rightarrow y \in \text{generate-field } K \Rightarrow x * y \in \text{generate-field } K$
 | *generate-field-inverse* [intro]: $x \in \text{generate-field } K \Rightarrow \text{inverse } x \in \text{generate-field } K$

lemma *subset-generate-field*: $A \subseteq \text{generate-field } A$
 ⟨proof⟩

26.2 The subfield locale

locale *Subfield* =
 fixes *K* :: 'a :: field set
 assumes *zero-closed* [intro, simp]: $0 \in K$
 assumes *one-closed* [intro, simp]: $1 \in K$
 assumes *add-closed* [intro]: $x \in K \Rightarrow y \in K \Rightarrow x + y \in K$
 assumes *uminus-closed* [intro]: $x \in K \Rightarrow -x \in K$
 assumes *mult-closed* [intro]: $x \in K \Rightarrow y \in K \Rightarrow x * y \in K$
 assumes *inverse-closed* [intro]: $x \in K \Rightarrow \text{inverse } x \in K$
begin

lemma *diff-closed* [intro]: $x \in K \Rightarrow y \in K \Rightarrow x - y \in K$
 ⟨proof⟩

lemma *divide-closed* [intro]: $x \in K \Rightarrow y \in K \Rightarrow x / y \in K$
 ⟨proof⟩

lemma *power-closed* [intro]: $x \in K \Rightarrow x ^ n \in K$
 ⟨proof⟩

lemma *of-nat-closed* [intro]: *of-nat* *n* $\in K$
 ⟨proof⟩

lemma *of-int-closed* [intro]: *of-int* *n* $\in K$
 ⟨proof⟩

lemma *sum-closed* [intro]: $(\bigwedge x. x \in A \Rightarrow f x \in K) \Rightarrow (\sum_{x \in A}. f x) \in K$
 ⟨proof⟩

lemma *prod-closed* [intro]: $(\bigwedge x. x \in A \Rightarrow f x \in K) \Rightarrow (\prod_{x \in A}. f x) \in K$
 ⟨proof⟩

lemma *cardK-gt1*:

assumes *finite K* **shows** $\text{card } K > 1$
 $\langle \text{proof} \rangle$

end

lemma *subfield-generate-field* [intro]:
fixes $A :: 'a :: \text{field set}$
shows *Subfield* (*generate-field* A)
 $\langle \text{proof} \rangle$

lemma *generate-field-least*:
assumes *Subfield* K $A \subseteq K$
shows *generate-field* $A \subseteq K$
 $\langle \text{proof} \rangle$

lemma *generate-field-mono*: $A \subseteq B \implies \text{generate-field } A \subseteq \text{generate-field } B$
 $\langle \text{proof} \rangle$

lemma *generate-field-mono'*: $A \subseteq B \implies x \in \text{generate-field } A \implies x \in \text{generate-field } B$
 $\langle \text{proof} \rangle$

lemma *generate-field-collapse* [simp]: *generate-field* (*generate-field* X) = *generate-field* X
 $\langle \text{proof} \rangle$

lemma *generate-field-Un-collapse1* [simp]:
generate-field (*generate-field* $A \cup B$) = *generate-field* ($A \cup B$)
 $\langle \text{proof} \rangle$

lemma *generate-field-Un-collapse2* [simp]:
generate-field ($A \cup \text{generate-field } B$) = *generate-field* ($A \cup B$)
 $\langle \text{proof} \rangle$

26.3 Field homomorphisms on a subfield

locale *field-hom-on* = *Subfield* K **for** $K :: 'a :: \text{field set} +$
fixes $f :: 'a \Rightarrow 'b :: \text{field}$
assumes *hom-0* [simp]: $f \ 0 = 0$
assumes *hom-1* [simp]: $f \ 1 = 1$
assumes *hom-add*: $x \in K \implies y \in K \implies f \ (x + y) = f \ x + f \ y$
assumes *hom-mult*: $x \in K \implies y \in K \implies f \ (x * y) = f \ x * f \ y$
begin

lemma *hom-uminus*: $x \in K \implies f \ (-x) = -f \ x$
 $\langle \text{proof} \rangle$

lemma *hom-diff*: $x \in K \implies y \in K \implies f \ (x - y) = f \ x - f \ y$
 $\langle \text{proof} \rangle$

lemma *hom-inverse*: $x \in K - \{0\} \implies f \text{ (inverse } x) = \text{inverse } (f x)$
 ⟨proof⟩

lemma *hom-inverse-all*: $x \in K \implies f \text{ (inverse } x) = \text{inverse } (f x)$
 ⟨proof⟩

lemma *hom-divide*: $x \in K \implies y \in K - \{0\} \implies f (x / y) = f x / f y$
 ⟨proof⟩

lemma *kernel-triv*:
assumes $u \in K - \{0\} f u = 0$
shows *False*
 ⟨proof⟩

A field homomorphism is injective on its domain.

lemma *inj-on*: $\text{inj-on } f K$
 ⟨proof⟩

lemma *generate-field-subK*: $S \subseteq K \implies \text{generate-field } S \subseteq K$
 ⟨proof⟩

The image under f of the subfield generated by a subset $S \subseteq K$ is contained in the subfield generated by the image $f ' S$: the homomorphism laws carry each generation step over (here f need only be a homomorphism on K , and $\text{generate-field } S \subseteq K$ keeps every intermediate element in range).

lemma *image-generate-field-subset*:
assumes $S: S \subseteq K$
shows $f ' \text{generate-field } S \subseteq \text{generate-field } (f ' S)$
 ⟨proof⟩

The image of a generated subfield is itself a subfield (of the target type).

lemma *image-generate-field-subfield*:
assumes $S: S \subseteq K$
shows $\text{Subfield } (f ' \text{generate-field } S)$
 ⟨proof⟩

Hence the image of a generated subfield is exactly the subfield generated by the image. (Both sides live in the target type, so this is an unconditional naturality statement.)

lemma *image-generate-field-eq*:
assumes $S: S \subseteq K$
shows $f ' \text{generate-field } S = \text{generate-field } (f ' S)$
 ⟨proof⟩

end

end

27 Polynomials over a Subfield; Minimal Polynomials in a Tower

```

theory Field-Extension-Tower
  imports Subfield
begin

```

For the iso-extension step of Galois theory we must reason about the minimal polynomial of an element over an *intermediate* subfield K (not just the prime field). We model a polynomial “over K ” as an *'a poly* all of whose coefficients lie in the subfield K . The key fact is that polynomial division by a *monic* polynomial over K keeps quotient and remainder over K ; everything then takes place inside the single principal ideal domain *'a poly*.

27.1 Polynomials with coefficients in a subfield

```

definition poly-over :: 'a :: field set  $\Rightarrow$  'a poly set where
  poly-over  $K \equiv \{p. \text{set } (\text{coeffs } p) \subseteq K\}$ 

```

```

lemma poly-over-iff-aux:
  assumes  $0 \in K$ 
  shows  $p \in \text{poly-over } K \iff (\forall i. \text{coeff } p \ i \in K)$ 
   $\langle \text{proof} \rangle$ 

```

```

context Subfield
begin

```

```

lemma poly-over-iff:  $p \in \text{poly-over } K \iff (\forall i. \text{coeff } p \ i \in K)$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma poly-over-0 [intro, simp]:  $0 \in \text{poly-over } K$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma poly-over-coeff:  $p \in \text{poly-over } K \implies \text{coeff } p \ i \in K$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma poly-over-I:  $(\bigwedge i. \text{coeff } p \ i \in K) \implies p \in \text{poly-over } K$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma poly-over-add:  $p \in \text{poly-over } K \implies q \in \text{poly-over } K \implies p + q \in \text{poly-over } K$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma poly-over-uminus:  $p \in \text{poly-over } K \implies -p \in \text{poly-over } K$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma poly-over-diff:  $p \in \text{poly-over } K \implies q \in \text{poly-over } K \implies p - q \in \text{poly-over } K$ 
   $\langle \text{proof} \rangle$ 

```

lemma *poly-over-smult*: $c \in K \implies p \in \text{poly-over } K \implies \text{smult } c \ p \in \text{poly-over } K$
 ⟨proof⟩

lemma *poly-over-monom*: $c \in K \implies \text{monom } c \ n \in \text{poly-over } K$
 ⟨proof⟩

lemma *poly-over-pCons*: $c \in K \implies p \in \text{poly-over } K \implies \text{pCons } c \ p \in \text{poly-over } K$
 ⟨proof⟩

lemma *poly-over-mult*:
assumes $p \in \text{poly-over } K$ $q \in \text{poly-over } K$ **shows** $p * q \in \text{poly-over } K$
 ⟨proof⟩

lemma *poly-over-1* [intro, simp]: $1 \in \text{poly-over } K$
 ⟨proof⟩

lemma *lead-coeff-closed*: $p \in \text{poly-over } K \implies \text{lead-coeff } p \in K$
 ⟨proof⟩

end

Enlarging the subfield enlarges the set of polynomials over it: this lets a tower $K \subseteq L$ of intermediate fields reuse the same ambient 'a poly.

lemma *poly-over-mono*: $K \subseteq L \implies \text{poly-over } K \subseteq \text{poly-over } L$
 ⟨proof⟩

27.2 Division by a monic polynomial over a subfield

Long division of a polynomial over K by a *monic* polynomial over K yields a quotient and remainder over K . We first show existence by induction on the degree, then identify the witnesses with the ambient (*div*) and (*mod*) by uniqueness of polynomial division.

lemma *monic-degree-0-eq-1*:
fixes $g :: 'a :: \text{field poly}$
assumes $\text{degree } g = 0$ $\text{lead-coeff } g = 1$ **shows** $g = 1$
 ⟨proof⟩

lemma *degree-drop-aux*:
fixes $q :: 'a :: \text{field poly}$
assumes $z: \text{coeff } q \ n = 0$ **and** $le: \text{degree } q \leq n$ **and** $npos: 0 < n$
shows $\text{degree } q < n$
 ⟨proof⟩

context *Subfield*
begin

lemma *poly-over-divmod-exists*:

fixes $d :: 'a \text{ poly}$
assumes $d: d \in \text{poly-over } K$ **and** *monic*: $\text{lead-coeff } d = 1$ **and** $ddeg: \text{degree } d > 0$
shows $p \in \text{poly-over } K \implies \exists q r. q \in \text{poly-over } K \wedge r \in \text{poly-over } K \wedge p = q * d + r \wedge \text{degree } r < \text{degree } d$
 $\langle \text{proof} \rangle$

By uniqueness of division, $p \text{ div } d$ and $p \text{ mod } d$ lie over K .

lemma *poly-over-div-mod*:
fixes $d :: 'a :: \text{field poly}$
assumes $d: d \in \text{poly-over } K$ **and** *monic*: $\text{lead-coeff } d = 1$ **and** $ddeg: \text{degree } d > 0$
and $p: p \in \text{poly-over } K$
shows $p \text{ div } d \in \text{poly-over } K \wedge p \text{ mod } d \in \text{poly-over } K$
 $\langle \text{proof} \rangle$

end

27.3 Minimal polynomial over a subfield

An element a is *algebraic over* K if some nonzero polynomial over K vanishes at it. A *minimal polynomial* is a monic polynomial over K of least positive degree vanishing at a .

definition *algebraic-over* $:: 'a :: \text{field set} \Rightarrow 'a \Rightarrow \text{bool}$ **where**
 $\text{algebraic-over } K a \equiv (\exists p \in \text{poly-over } K. p \neq 0 \wedge \text{poly } p a = 0)$

definition *is-minpoly* $:: 'a :: \text{field set} \Rightarrow 'a \Rightarrow 'a \text{ poly} \Rightarrow \text{bool}$ **where**
 $\text{is-minpoly } K a m \equiv$
 $m \in \text{poly-over } K \wedge \text{lead-coeff } m = 1 \wedge \text{poly } m a = 0 \wedge$
 $(\forall r \in \text{poly-over } K. r \neq 0 \wedge \text{poly } r a = 0 \longrightarrow \text{degree } m \leq \text{degree } r)$

A polynomial is *irreducible over* K if it has positive degree and admits no factorisation over K into two factors of positive degree. (Over the ambient field it may well factor — e.g. split completely — but not within *poly-over* K .)

definition *irreducible-over* $:: 'a :: \text{field set} \Rightarrow 'a \text{ poly} \Rightarrow \text{bool}$ **where**
 $\text{irreducible-over } K p \equiv$
 $p \neq 0 \wedge \text{degree } p > 0 \wedge$
 $(\forall b c. b \in \text{poly-over } K \longrightarrow c \in \text{poly-over } K \longrightarrow p = b * c \longrightarrow \text{degree } b = 0 \vee \text{degree } c = 0)$

Algebraicity is preserved when passing to a larger subfield.

lemma *algebraic-over-mono*:
assumes $\text{algebraic-over } K a$ **and** $K \subseteq L$
shows $\text{algebraic-over } L a$
 $\langle \text{proof} \rangle$

context *Subfield*
begin

A minimal polynomial exists for every algebraic element: take a nonzero annihilator of least degree and normalise it to be monic (the normalising scalar lies in K).

lemma *minpoly-exists*:
assumes *algebraic-over* K a
shows $\exists m. \text{is-minpoly } K \ a \ m$
 $\langle \text{proof} \rangle$

The kernel characterization: any polynomial over K vanishing at a is a multiple of the minimal polynomial. (Divide by m : the remainder is a lower-degree annihilator over K , hence zero by minimality.)

lemma *minpoly-dvd*:
assumes $m: \text{is-minpoly } K \ a \ m$ **and** $r: r \in \text{poly-over } K$ **and** $ra: \text{poly } r \ a = 0$
shows $m \text{ dvd } r$
 $\langle \text{proof} \rangle$

The minimal polynomial has positive degree and is irreducible over K : a proper factorisation over K would give a nonzero factor of smaller degree annihilating a , contradicting minimality.

lemma *minpoly-degree-pos*:
assumes $m: \text{is-minpoly } K \ a \ m$ **shows** $\text{degree } m > 0$
 $\langle \text{proof} \rangle$

lemma *minpoly-no-proper-factor*:
assumes $m: \text{is-minpoly } K \ a \ m$
and $b: b \in \text{poly-over } K$ **and** $c: c \in \text{poly-over } K$ **and** $eq: m = b * c$
shows $\text{degree } b = 0 \vee \text{degree } c = 0$
 $\langle \text{proof} \rangle$

B
 'ezout over K : if m is the (irreducible) minimal polynomial and m does not divide a polynomial p over K , then 1 is a K -coefficient combination of m and p . Proved via a least-degree element of the combination set, which (by irreducibility of m) must be a unit.

lemma *minpoly-bezout*:
assumes $m: \text{is-minpoly } K \ a \ m$ **and** $p: p \in \text{poly-over } K$ **and** $ndvd: \neg m \text{ dvd } p$
shows $\exists u \in \text{poly-over } K. \exists v \in \text{poly-over } K. u * m + v * p = 1$
 $\langle \text{proof} \rangle$

If p is irreducible over K and vanishes at a , then p is an associate (over K) of the minimal polynomial of a : the minimal polynomial divides p , and irreducibility forces the cofactor to be a nonzero constant.

lemma *irreducible-over-minpoly-assoc*:
assumes $\text{irr}: \text{irreducible-over } K \ p$ **and** $m: \text{is-minpoly } K \ a \ m$
and $pa: \text{poly } p \ a = 0$ **and** $pK: p \in \text{poly-over } K$
shows $\exists k. k \neq 0 \wedge p = \text{smult } k \ m$
 $\langle \text{proof} \rangle$

Consequently every root of an irreducible p is a root of the minimal polynomial of any other root: this is what makes the minimal-polynomial hypothesis of the transitivity theorem automatic for irreducible polynomials.

lemma *irreducible-over-root-is-minpoly-root*:

assumes *irreducible-over* K p $p \in \text{poly-over } K$ *is-minpoly* K a m *poly* p $a = 0$ *poly* p $b = 0$
shows *poly* m $b = 0$
 $\langle \text{proof} \rangle$

end

end

28 Simple Algebraic Extensions are Subfields

theory *Simple-Extension*

imports *Field-Extension-Tower*

begin

The simple extension $K(a)$ of a subfield K by an element a algebraic over K is realised as the set of polynomial expressions *poly* p a for $p \in \text{poly-over } K$. It is a subfield: closure under the ring operations is immediate (evaluation is a ring homomorphism), and closure under inverse is the payoff of Bézout's identity over K .

definition *eval-img* :: ' a :: field set $\Rightarrow$ ' $a \Rightarrow$ ' a set **where**

eval-img K $a = (\lambda p. \text{poly } p \ a) \text{ 'poly-over } K$

lemma *eval-imgI*: $p \in \text{poly-over } K \implies \text{poly } p \ a \in \text{eval-img } K \ a$

$\langle \text{proof} \rangle$

lemma *eval-imgE*:

assumes $y \in \text{eval-img } K \ a$ **obtains** p **where** $p \in \text{poly-over } K$ $y = \text{poly } p \ a$

$\langle \text{proof} \rangle$

context *Subfield*

begin

lemma *eval-img-self*: $a \in \text{eval-img } K \ a$

$\langle \text{proof} \rangle$

lemma *eval-img-base*:

assumes $c \in K$

shows $c \in \text{eval-img } K \ a$

$\langle \text{proof} \rangle$

lemma *eval-img-0*: $0 \in \text{eval-img } K \ a$

$\langle \text{proof} \rangle$

lemma *eval-img-1*: $1 \in \text{eval-img } K \ a$
 ⟨proof⟩

lemma *eval-img-add*: $x \in \text{eval-img } K \ a \implies y \in \text{eval-img } K \ a \implies x + y \in \text{eval-img } K \ a$
 ⟨proof⟩

lemma *eval-img-uminus*: $x \in \text{eval-img } K \ a \implies -x \in \text{eval-img } K \ a$
 ⟨proof⟩

lemma *eval-img-mult*: $x \in \text{eval-img } K \ a \implies y \in \text{eval-img } K \ a \implies x * y \in \text{eval-img } K \ a$
 ⟨proof⟩

Closure under inverse, the payoff of Bézout.

lemma *eval-img-inverse*:
assumes *alg*: algebraic-over $K \ a$ **and** *y*: $y \in \text{eval-img } K \ a$ **and** *ynz*: $y \neq 0$
shows *inverse* $y \in \text{eval-img } K \ a$
 ⟨proof⟩

Hence, for a algebraic over K , the simple extension is a subfield.

theorem *subfield-eval-img*:
assumes algebraic-over $K \ a$ **shows** *Subfield* ($\text{eval-img } K \ a$)
 ⟨proof⟩

The simple extension $\text{eval-img } K \ a$ coincides with the generically generated subfield *generate-field* ($K \cup \{a\}$) of theory *Subfield*. This connects the polynomial-evaluation construction to the inductive *generate-field*, so the tower of intermediate fields can be expressed either way. The forward inclusion is unconditional; the reverse uses that $\text{eval-img } K \ a$ is itself a subfield (so *generate-field* of a generating subset of it stays inside).

lemma *eval-img-subset-generate-field*:
assumes *p* $\in \text{poly-over } K$
shows *poly* $p \ a \in \text{generate-field } (K \cup \{a\})$
 ⟨proof⟩

lemma *eval-img-eq-generate-field*:
assumes *alg*: algebraic-over $K \ a$
shows $\text{eval-img } K \ a = \text{generate-field } (K \cup \{a\})$
 ⟨proof⟩

end

end

29 Extending a Field Homomorphism across a Simple Algebraic Extension

```

theory Iso-Extension
  imports Simple-Extension
begin

```

The keystone of the iso-extension tower. Let f be a homomorphism defined on a subfield K (a *field-hom-on*), let a be algebraic over K with minimal polynomial m , and let b be a root of the f -image $\text{map-poly } f \ m$ of that minimal polynomial. Then f extends to a homomorphism g on the simple extension $\text{eval-img } K \ a (= K(a))$ that sends a to b and restricts to f on K .

The extension is forced: every element of $K \ a$ is $\text{poly } p \ a$ for some $p \in \text{poly-over } K$, and g must send it to $\text{poly } (\text{map-poly } f \ p) \ b$. Well-definedness is the only subtlety: if $\text{poly } p \ a = \text{poly } q \ a$ then m divides $p - q$ (the kernel characterization *minpoly-dvd*), so $\text{map-poly } f \ m$ divides $\text{map-poly } f \ (p - q)$, which therefore vanishes at its root b .

```

context field-hom-on
begin

```

29.1 The mapped polynomial $\text{map-poly } f$ on $\text{poly-over } K$

Since f is only a homomorphism *on* K , the library's *map-poly* homomorphism lemmas (which assume a global ring homomorphism) do not apply. We re-establish additivity and multiplicativity of *map-poly* f restricted to polynomials over K , working coefficientwise.

```

lemma hom-sum:
  assumes  $\bigwedge i. i \in A \implies x \ i \in K$ 
  shows  $f \ (\text{sum } x \ A) = (\sum i \in A. f \ (x \ i))$ 
  <proof>

```

The coefficient lemma uses only the global *hom-0*.

```

lemma coeff-map-poly-f:  $\text{coeff } (\text{map-poly } f \ p) \ n = f \ (\text{coeff } p \ n)$ 
  <proof>

```

```

lemma map-poly-add-on:
  assumes  $p \in \text{poly-over } K \ q \in \text{poly-over } K$ 
  shows  $\text{map-poly } f \ (p + q) = \text{map-poly } f \ p + \text{map-poly } f \ q$ 
  <proof>

```

```

lemma map-poly-mult-on:
  assumes  $p \in \text{poly-over } K \ q \in \text{poly-over } K$ 
  shows  $\text{map-poly } f \ (p * q) = \text{map-poly } f \ p * \text{map-poly } f \ q$ 
  <proof>

```

```

lemma map-poly-diff-on:
  assumes  $p \in \text{poly-over } K \ q \in \text{poly-over } K$ 

```

shows $\text{map-poly } f \ (p - q) = \text{map-poly } f \ p - \text{map-poly } f \ q$
 $\langle \text{proof} \rangle$

lemma map-poly-const : $\text{map-poly } f \ [:c:] = [:f \ c:]$
 $\langle \text{proof} \rangle$

29.2 Evaluation commutes with the homomorphism

f respects powers of elements of K .

lemma hom-power : $x \in K \implies f \ (x \wedge n) = (f \ x) \wedge n$
 $\langle \text{proof} \rangle$

For a polynomial over K and an argument in K , evaluating the f -image at $f \ x$ equals applying f to the value at x : evaluation is natural in f . (The degree of $\text{map-poly } f \ p$ may drop if f kills the leading coefficient, hence the *sum.mono-neutral-left* step.)

lemma poly-map-poly-hom :
assumes p : $p \in \text{poly-over } K$ **and** x : $x \in K$
shows $\text{poly} \ (\text{map-poly } f \ p) \ (f \ x) = f \ (\text{poly } p \ x)$
 $\langle \text{proof} \rangle$

Consequently f sends a root in K of a polynomial over K to a root of the mapped polynomial.

lemma maps-root :
assumes p : $p \in \text{poly-over } K$ **and** x : $x \in K$ **and** root : $\text{poly } p \ x = 0$
shows $\text{poly} \ (\text{map-poly } f \ p) \ (f \ x) = 0$
 $\langle \text{proof} \rangle$

29.3 Well-definedness of the extension

If two polynomials over K agree at a , their f -images agree at any root b of $\text{map-poly } f \ m$. This is what makes the extension a function.

lemma eval-unique :
assumes m : $\text{is-minpoly } K \ a \ m$ **and** b : $\text{poly} \ (\text{map-poly } f \ m) \ b = 0$
and p : $p \in \text{poly-over } K$ **and** q : $q \in \text{poly-over } K$ **and** eq : $\text{poly } p \ a = \text{poly } q \ a$
shows $\text{poly} \ (\text{map-poly } f \ p) \ b = \text{poly} \ (\text{map-poly } f \ q) \ b$
 $\langle \text{proof} \rangle$

29.4 The extension theorem

theorem iso-extension :
assumes alg : $\text{algebraic-over } K \ a$
and m : $\text{is-minpoly } K \ a \ m$
and b : $\text{poly} \ (\text{map-poly } f \ m) \ b = 0$
shows $\exists g. \text{field-hom-on} \ (\text{eval-img } K \ a) \ g \wedge g \ a = b \wedge (\forall x \in K. g \ x = f \ x)$
 $\langle \text{proof} \rangle$

end

end

```
theory Ring-Theory
  imports Group-Theory
begin
```

```
no-notation plus (infixl <+> 65)
no-notation minus (infixl <-> 65)
unbundle no uminus-syntax
no-notation quotient (infixl <'/'> 90)
```

30 Rings

30.1 Definition and Elementary Properties

Def 2.1

p 86, ll 20–28

```
locale Ring = additive: Abelian-Group R (+) 0 + multiplicative: Monoid R (·) 1
  for R and addition (infixl <+> 65) and multiplication (infixl <·> 70) and zero
  (<0>) and unit (<1>) +
  assumes distributive:  $\llbracket a \in R; b \in R; c \in R \rrbracket \implies a \cdot (b + c) = a \cdot b + a \cdot c$ 
     $\llbracket a \in R; b \in R; c \in R \rrbracket \implies (b + c) \cdot a = b \cdot a + c \cdot a$ 
begin
```

p 86, ll 20–28

```
notation additive.inverse (<- -> [66] 65)
abbreviation subtraction (infixl <-> 65) where  $a - b \equiv a + (- b)$ 
```

end

Compact version based on the monoid type

```
locale Ring' =
  fixes additive multiplicative
  assumes eq: mcarrier additive = mcarrier multiplicative
  assumes Ring: Ring (mcarrier additive) (mmult additive) (mmult multiplicative)
  (munit additive) (munit multiplicative)
```

p 87, ll 10–12

```
locale Subring =
  additive: Subgroup S R (+) 0 + multiplicative: Submonoid S R (·) 1
  for S and R and addition (infixl <+> 65) and multiplication (infixl <·> 70) and
  zero (<0>) and unit (<1>)
```

```
context Ring begin
```

p 88, ll 26–28

```
lemma right-zero [simp]:  
  assumes [simp]:  $a \in R$  shows  $a \cdot 0 = 0$   
   $\langle proof \rangle$ 
```

p 88, l 29

```
lemma left-zero [simp]:  
  assumes [simp]:  $a \in R$  shows  $0 \cdot a = 0$   
   $\langle proof \rangle$ 
```

p 88, ll 29–30; p 89, ll 1–2

```
lemma left-minus:  
  assumes [simp]:  $a \in R$   $b \in R$  shows  $(- a) \cdot b = - a \cdot b$   
   $\langle proof \rangle$ 
```

p 89, l 3

```
lemma right-minus:  
  assumes [simp]:  $a \in R$   $b \in R$  shows  $a \cdot (- b) = - a \cdot b$   
   $\langle proof \rangle$ 
```

end

30.2 Commutative rings, integral domains, and fields

A commutative ring is a ring whose multiplication is commutative. We reuse the *commutative-monoid* structure on the multiplicative part, mirroring how *Abelian-Group* reuses *commutative-monoid*.

```
locale commutative-ring = Ring +  
  multiplicative: commutative-monoid  $R$   $(\cdot)$  1  
begin
```

Associative-commutative normalisation bundles for the two operations, for on-demand use as *simp add: mult-ac* / *simp add: add-ac*. The additive group is abelian, so it too supplies the *ac* bundle.

```
lemmas mult-ac = multiplicative.ac  
lemmas add-ac = additive.ac
```

end

A nontrivial ring is one in which the unit and zero differ. Both integral domains and fields require this, so we factor it into a common ancestor: that way the assumption is inherited (and named *nontrivial*) by a single route, avoiding a duplicate when a field is shown to be an integral domain or interpreted.

```
locale nontrivial-ring = Ring +  
  assumes nontrivial:  $1 \neq 0$ 
```

An integral domain is a nontrivial commutative ring without zero divisors.

locale *integral-domain* = *commutative-ring* + *nontrivial-ring* +
assumes *no-zero-divisors*: $\llbracket a \in R; b \in R; a \cdot b = \mathbf{0} \rrbracket \implies a = \mathbf{0} \vee b = \mathbf{0}$

A field is a nontrivial commutative ring in which every nonzero element is invertible. Invertibility is expressed through the multiplicative monoid's *Monoid.invertible* predicate, so the multiplicative inverse and unit machinery is available via the *multiplicative* prefix.

locale *Field* = *commutative-ring* + *nontrivial-ring* +
assumes *field-inverse*: $\llbracket a \in R; a \neq \mathbf{0} \rrbracket \implies \text{multiplicative.invertible } a$
begin

Every field is an integral domain.

sublocale *integral-domain*
 $\langle \text{proof} \rangle$

end

30.3 Ideals, Quotient Rings

p 101, ll 2–5

locale *ring-congruence* = *Ring* +
additive: *group-congruence* *R* (+) **0** *E* +
multiplicative: *Monoid-congruence* *R* ($\cdot$) **1** *E*
for *E*
begin

p 101, ll 2–5

notation *additive.quotient-composition* (**infixl** $\langle [+]\rangle$ 65)
notation *additive.quotient.inverse* ($\langle [-] \rightarrow [66] \rangle$ 65)
notation *multiplicative.quotient-composition* (**infixl** $\langle [\cdot] \rangle$ 70)

p 101, ll 5–11

sublocale *quotient*: *Ring* *R* / *E* ([+]) ([$\cdot$]) *additive.Class* **0** *additive.Class* **1**
 $\langle \text{proof} \rangle$

end

p 101, ll 12–13

locale *subgroup-of-additive-group-of-ring* =
additive: *Subgroup* *I* *R* (+) **0** + *Ring* *R* (+) ($\cdot$) **0** **1**
for *I* **and** *R* **and** *addition* (**infixl** $\langle +\rangle$ 65) **and** *multiplication* (**infixl** $\langle \cdot \rangle$ 70) **and**
zero ($\langle \mathbf{0} \rangle$) **and** *unit* ($\langle \mathbf{1} \rangle$)
begin

p 101, ll 13–14

definition *Ring-Congruence* = $\{(a, b). a \in R \wedge b \in R \wedge a - b \in I\}$

p 101, ll 13–14

lemma *Ring-CongruenceI*: $\llbracket a - b \in I; a \in R; b \in R \rrbracket \implies (a, b) \in \text{Ring-Congruence}$
<proof>

p 101, ll 13–14

lemma *Ring-CongruenceD*: $(a, b) \in \text{Ring-Congruence} \implies a - b \in I$
<proof>

Jacobson’s definition of Ring congruence deviates from that of group congruence;
this complicates the proof.

p 101, ll 12–14

sublocale *additive: subgroup-of-abelian-group I R (+) 0*
rewrites *additive-congruence: additive.Congruence = Ring-Congruence*
<proof>

p 101, l 14

notation *additive.Left-Coset* (**infixl** $\langle + | \rangle$ 65)

end

Def 2.2

p 101, ll 21–22

locale *Ideal = subgroup-of-additive-group-of-ring +*
assumes *Ideal*: $\llbracket a \in R; b \in I \rrbracket \implies a \cdot b \in I \llbracket a \in R; b \in I \rrbracket \implies b \cdot a \in I$

context *subgroup-of-additive-group-of-ring* **begin**

p 101, ll 14–17

theorem *multiplicative-congruence-implies-ideal*:
assumes *Monoid-congruence R* $(\cdot)$ **1** *Ring-Congruence*
shows *Ideal I R* $(+)$ $(\cdot)$ **0 1**
<proof>

end

context *Ideal* **begin**

p 101, ll 17–20

theorem *multiplicative-congruence [intro]*:
assumes *a*: $(a, a') \in \text{Ring-Congruence}$ **and** *b*: $(b, b') \in \text{Ring-Congruence}$
shows $(a \cdot b, a' \cdot b') \in \text{Ring-Congruence}$
<proof>

p 101, ll 23–24

sublocale *ring-congruence* **where** $E = \text{Ring-Congruence}$

$\langle proof \rangle$

end

p 101, ll 24–26

locale *quotient-ring* = *Ideal* **begin**

p 101, ll 24–26

sublocale *quotient*: *Ring* *R* / (*subgroup-of-additive-group-of-ring.Ring-Congruence* *I* *R* (+) **0**) ([+]) ([·]) *additive.Class* **0** *additive.Class* **1** $\langle proof \rangle$

p 101, l 26

lemmas *Left-Coset* = *additive.Left-CosetE*

Equation 17 (1)

p 101, l 28

lemmas *quotient-addition* = *additive.factor-composition*

Equation 17 (2)

p 101, l 29

theorem *quotient-multiplication* [simp]:
 $\llbracket a \in R; b \in R \rrbracket \implies (a + I) [\cdot] (b + I) = a \cdot b + I$
 $\langle proof \rangle$

p 101, l 30

lemmas *quotient-zero* = *additive.factor-unit*

lemmas *quotient-negative* = *additive.factor-inverse*

end

30.4 Homomorphisms of Rings. Basic Theorems

Def 2.3

p 106, ll 7–9

locale *ring-homomorphism* =
map η *R* *R'* + *source*: *Ring* *R* (+) (·) **0** **1** + *target*: *Ring* *R'* (+') (·') **0'** **1'** +
additive: *group-homomorphism* η *R* (+) **0** *R'* (+') **0'** +
multiplicative: *Monoid-homomorphism* η *R* (·) **1** *R'* (·') **1'**
for η
and *R* **and** *addition* (infixl $\langle + \rangle$ 65) **and** *multiplication* (infixl $\langle \cdot \rangle$ 70) **and** *zero*
($\langle 0 \rangle$) **and** *unit* ($\langle 1 \rangle$)
and *R'* **and** *addition'* (infixl $\langle +' \rangle$ 65) **and** *multiplication'* (infixl $\langle \cdot' \rangle$ 70) **and**
zero' ($\langle 0' \rangle$) **and** *unit'* ($\langle 1' \rangle$)

p 106, l 17

locale *ring-epimorphism* = *ring-homomorphism* + *surjective-map* η R R'

p 106, ll 14–18

sublocale *quotient-ring* $\subseteq$ *natural: ring-epimorphism*
where η = *additive.Class* **and** $R' = R / (\text{subgroup-of-additive-group-of-ring}.\text{Ring-Congruence}$
 I R $(+)$ **0**) **and** *addition'* = $([+])$ **and** *multiplication'* = $([.])$
and *zero'* = *additive.Class* **0** **and** *unit'* = *additive.Class* **1**
 $\langle \text{proof} \rangle$

context *ring-homomorphism* **begin**

Jacobson reasons via $a - b \in \text{additive.Ker}$ being a congruence; we prefer the direct proof, since it is very simple.

p 106, ll 19–21

sublocale *kernel: Ideal* **where** $I = \text{additive.Ker}$
 $\langle \text{proof} \rangle$

end

p 106, l 22

locale *ring-monomorphism* = *ring-homomorphism* + *injective-map* η R R'

context *ring-homomorphism* **begin**

p 106, ll 21–23

theorem *ring-monomorphism-iff-kernel-unit:*
ring-monomorphism η R $(+)$ $(\cdot)$ **0** **1** R' $(+)$ $(\cdot)$ **0'** **1'** $\longleftrightarrow \text{additive.Ker} = \{0\}$ **(is**
 $?monom \longleftrightarrow ?ker)$
 $\langle \text{proof} \rangle$

end

p 106, ll 23–25

sublocale *ring-homomorphism* $\subseteq$ *image: Subring* η R R' $(+)$ $(\cdot)$ **0'** **1'** $\langle \text{proof} \rangle$

p 106, ll 26–27

locale *ideal-in-kernel* =
ring-homomorphism + *contained: Ideal* I R $(+)$ $(\cdot)$ **0** **1** **for** I +
assumes *subset: $I \subseteq \text{additive.Ker}$*
begin

p 106, ll 26–27

notation *contained.additive.quotient-composition* (**infixl** $\langle [+]$ 65)

notation *contained.multiplicative.quotient-composition* (**infixl** $\langle [.]$ 70)

Provides *additive.induced*, which Jacobson calls $\bar{\eta}$.

p 106, l 30

sublocale *additive: normal-subgroup-in-kernel* $\eta \ R \ (+) \ \mathbf{0} \ R' \ (+') \ \mathbf{0}' \ I$
rewrites *normal-subgroup.Congruence* $I \ R \ addition \ zero = contained.Ring-Congruence$
<proof>

Only the multiplicative part needs some work.

p 106, ll 27–30

sublocale *induced: ring-homomorphism* *additive.induced* $R \ / \ (subgroup-of-additive-group-of-ring.Ring-Congruence$
 $I \ R \ (+) \ \mathbf{0} \ ([+]) \ ([\cdot]) \ contained.additive.Class \ \mathbf{0} \ contained.additive.Class \ \mathbf{1}$
<proof>

p 106, l 30; p 107, ll 1–3

additive.induced denotes Jacobson's $\bar{\eta}$. We have the commutativity of the diagram, where *additive.induced* is unique:

compose R additive.induced contained.additive.Class = η

$\llbracket ?\beta \in contained.additive.Partition \rightarrow_E R';$
compose R ? β contained.additive.Class = η
 $\implies ?\beta = additive.induced$

end

Fundamental Theorem of Homomorphisms of Rings

p 107, l 6

locale *ring-homomorphism-fundamental* = *ring-homomorphism* **begin**

p 107, l 6

notation *kernel.additive.quotient-composition* (**infixl** $\langle [+]\rangle$ 65)
notation *kernel.multiplicative.quotient-composition* (**infixl** $\langle [\cdot]\rangle$ 70)

p 107, l 6

sublocale *ideal-in-kernel* **where** $I = additive.Ker$ *<proof>*

p 107, ll 8–9

sublocale *natural: ring-epimorphism*
where $\eta = kernel.additive.Class$ **and** $R' = R \ / \ (subgroup-of-additive-group-of-ring.Ring-Congruence$
 $additive.Ker \ R \ (+) \ \mathbf{0})$
and $addition' = kernel.additive.quotient-composition$
and $multiplication' = kernel.multiplicative.quotient-composition$
and $zero' = kernel.additive.Class \ \mathbf{0}$ **and** $unit' = kernel.additive.Class \ \mathbf{1}$
<proof>

p 107, l 9

sublocale *induced: ring-monomorphism*

where $\eta = \text{additive.induced}$ **and** $R = R / (\text{subgroup-of-additive-group-of-ring.Ring-Congruence additive.Ker } R (+) \mathbf{0})$
and $\text{addition} = \text{kernel.additive.quotient-composition}$
and $\text{multiplication} = \text{kernel.multiplicative.quotient-composition}$
and $\text{zero} = \text{kernel.additive.Class } \mathbf{0}$ **and** $\text{unit} = \text{kernel.additive.Class } \mathbf{1}$
 $\langle \text{proof} \rangle$

end

p 107, l 11

locale $\text{ring-isomorphism} = \text{ring-homomorphism} + \text{bijective-map } \eta \ R \ R'$ **begin**

p 107, l 11

sublocale ring-monomorphism $\langle \text{proof} \rangle$

sublocale ring-epimorphism $\langle \text{proof} \rangle$

p 107, l 11

lemma $\text{inverse-ring-isomorphism}$:

$\text{ring-isomorphism } (\text{restrict } (\text{inv-into } R \ \eta) \ R') \ R' (+') (\cdot') \mathbf{0}' \mathbf{1}' R (+) (\cdot) \mathbf{0} \mathbf{1}$
 $\langle \text{proof} \rangle$

end

p 107, l 11

definition $\text{isomorphic-as-rings}$ (**infixl** $\langle \cong_R \rangle$ 50)

where $\mathcal{R} \cong_R \mathcal{R}' \longleftrightarrow (\text{let } (R, \text{addition}, \text{multiplication}, \text{zero}, \text{unit}) = \mathcal{R}; (R', \text{addition}', \text{multiplication}', \text{zero}', \text{unit}') = \mathcal{R}' \text{ in}$

$(\exists \eta. \text{ring-isomorphism } \eta \ R \ \text{addition} \ \text{multiplication} \ \text{zero} \ \text{unit} \ R' \ \text{addition}' \ \text{multiplication}' \ \text{zero}' \ \text{unit}'))$

p 107, l 11

lemma $\text{isomorphic-as-rings-symmetric}$:

$(R, \text{addition}, \text{multiplication}, \text{zero}, \text{unit}) \cong_R (R', \text{addition}', \text{multiplication}', \text{zero}', \text{unit}') \implies$

$(R', \text{addition}', \text{multiplication}', \text{zero}', \text{unit}') \cong_R (R, \text{addition}, \text{multiplication}, \text{zero}, \text{unit})$

$\langle \text{proof} \rangle$

context ring-homomorphism **begin**

Corollary

p 107, ll 11–12

theorem $\text{image-is-isomorphic-to-quotient-ring}$:

$\exists K \ \text{add} \ \text{mult} \ \text{zero} \ \text{one}. \text{Ideal } K \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \wedge (\eta \text{ ' } R, (+'), (\cdot'), \mathbf{0}', \mathbf{1}') \cong_R (R / (\text{subgroup-of-additive-group-of-ring.Ring-Congruence } K \ R \ (+) \ \mathbf{0}), \text{add}, \text{mult}, \text{zero}, \text{one})$
 $\langle \text{proof} \rangle$

end

30.5 Abstract type of rings

lemma *trivial-Ring'*: *Ring'* *trivial-monoid trivial-monoid*
 $\langle \text{proof} \rangle$

lemma *trivial-Ring*: *Ring* {*undefined*} ($\lambda x y. \text{undefined}$) ($\lambda x y. \text{undefined}$) *undefined*
 $\langle \text{proof} \rangle$

typedef 'a *ring* = {(*addition*::'a *monoid*, *multiplication*). *Ring'* *addition multiplication*}
morphisms *dest-ring ring*
 $\langle \text{proof} \rangle$

declare *dest-ring-inverse* [simp]

definition *trivial-ring* $\equiv \text{ring } (\text{trivial-monoid}, \text{trivial-monoid})$

definition *radd* **where** *radd m* $\equiv \text{fst } (\text{dest-ring } m)$

definition *rmult* **where** *rmult m* $\equiv \text{snd } (\text{dest-ring } m)$

definition *rcarrier* **where** *rcarrier m* $\equiv \text{mcarrier } (\text{radd } m)$

lemma *ring-is-Ring* [iff]: *Ring'* (*radd m*) (*rmult m*)
 $\langle \text{proof} \rangle$

lemma *ring-collapse* [simp]: *ring* (*radd r*, *rmult r*) = *r*
 $\langle \text{proof} \rangle$

Allows reference to the current ring space within the locale as a value

definition (in *Ring'*) *Self* $\equiv \text{ring } (\text{additive}, \text{multiplicative})$

lemma (in *Ring'*) *radd-Self* [simp]: *radd Self* = *additive*
 $\langle \text{proof} \rangle$

lemma (in *Ring'*) *rmult-Self* [simp]: *rmult Self* = *multiplicative*
 $\langle \text{proof} \rangle$

Further accessors, mirroring the 'a *monoid* type: the additive and multiplicative operations and the two distinguished constants **0**, **1** recovered from the component monoids. (We keep *radd*/*rmult* returning whole 'a *monoid* values, as *Ring'* is built on those.)

definition *radd-op* **where** *radd-op m* $\equiv \text{mmult } (\text{radd } m)$

definition *rmult-op* **where** *rmult-op m* $\equiv \text{mmult } (\text{rmult } m)$

definition *rzero* **where** *rzero m* $\equiv \text{munit } (\text{radd } m)$

definition *rone* **where** *rone m* $\equiv \text{munit } (\text{rmult } m)$

The value really is a ring: the analogue of *monoid-is-Monoid*.

lemma *Ring-ring [iff]: Ring (rcarrier m) (radd-op m) (rmult-op m) (rzero m) (rone m)*
⟨proof⟩

Computing the accessors of a ring built from its component monoids.

lemma (in *Ring'*) *radd-ring [simp]: radd (ring (additive, multiplicative)) = additive*
⟨proof⟩

lemma (in *Ring'*) *rmult-ring [simp]: rmult (ring (additive, multiplicative)) = multiplicative*
⟨proof⟩

lemma (in *Ring'*) *rcarrier-ring [simp]: rcarrier (ring (additive, multiplicative)) = mcarrier additive*
⟨proof⟩

lemma (in *Ring'*) *rzero-ring [simp]: rzero (ring (additive, multiplicative)) = munit additive*
⟨proof⟩

lemma (in *Ring'*) *rone-ring [simp]: rone (ring (additive, multiplicative)) = munit multiplicative*
⟨proof⟩

lemma (in *Ring'*) *radd-op-ring [simp]: radd-op (ring (additive, multiplicative)) = mmult additive*
⟨proof⟩

lemma (in *Ring'*) *rmult-op-ring [simp]: rmult-op (ring (additive, multiplicative)) = mmult multiplicative*
⟨proof⟩

Within the mathematical *Ring* locale, the component monoids form a *Ring'*, so the current ring can be referred to as a value.

lemma (in *Ring*) *Ring'-Self-components:*
Ring' (monoid (R, (+), 0)) (monoid (R, (·), 1))
⟨proof⟩

definition (in *Ring*) *Self* $\equiv$ *ring (monoid (R, (+), 0), monoid (R, (·), 1))*

lemma (in *Ring*) *rcarrier-Self [simp]: rcarrier Self = R*
⟨proof⟩

lemma (in *Ring*) *rzero-Self [simp]: rzero Self = 0*
⟨proof⟩

lemma (in *Ring*) *rone-Self [simp]: rone Self = 1*
⟨proof⟩

lemma (in *Ring*) *radd-op-Self* [simp]: *radd-op Self* = (+)
 ⟨proof⟩

lemma (in *Ring*) *rmult-op-Self* [simp]: *rmult-op Self* = (·)
 ⟨proof⟩

The one abstract *'a ring* type serves all the ring-like structures: which one a value belongs to is recorded by a predicate on its accessors. (We provide the field predicate, used in Galois theory; the others follow the same shape.)

definition *is-field* where

is-field *r* $\equiv$ *Field* (*rcarrier* *r*) (*radd-op* *r*) (*rmult-op* *r*) (*rzero* *r*) (*rone* *r*)

lemma *is-fieldI*:

Field (*rcarrier* *r*) (*radd-op* *r*) (*rmult-op* *r*) (*rzero* *r*) (*rone* *r*) $\implies$ *is-field* *r*
 ⟨proof⟩

lemma *is-fieldD*:

is-field *r* $\implies$ *Field* (*rcarrier* *r*) (*radd-op* *r*) (*rmult-op* *r*) (*rzero* *r*) (*rone* *r*)
 ⟨proof⟩

30.6 Linking the locale-based field theory to Isabelle's type classes

Following the *Met-TC* pattern of *HOL–Analysis.Abstract-Metric-Spaces* (where interpretation *Met-TC*: *Metric-space UNIV dist* feeds the type-class operations into the locale), we interpret the locale-based *Field* at an arbitrary type-class field, with the whole type as carrier. This lets the locale-based constructions to come (polynomials, splitting fields, Galois groups) be applied to concrete type-class fields such as *real* and *complex*.

The additive-group, multiplicative-monoid and ring bridge lemmas need only the ring axioms, so we state them at the weakest suitable type class *comm-ring-1* rather than *field*. This lets the same bridge serve integral domains (e.g. *int*, in theory *IntRing*), not just fields.

lemma *add-group-TC*: *Group* (*UNIV* :: '*a* :: group-add set) plus 0
 ⟨proof⟩

lemma *add-abelian-TC*: *Abelian-Group* (*UNIV* :: '*a* :: ab-group-add set) plus 0
 ⟨proof⟩

lemma *mult-monoid-TC*: *Monoid* (*UNIV* :: '*a* :: monoid-mult set) (*) 1
 ⟨proof⟩

lemma *mult-cmonoid-TC*: *commutative-monoid* (*UNIV* :: '*a* :: comm-monoid-mult set) (*) 1
 ⟨proof⟩

lemma *ring-TC*: *Ring* (*UNIV* :: 'a :: ring-1 set) *plus times 0 1*
 <proof>

lemma *comm-ring-TC*: *commutative-ring* (*UNIV* :: 'a :: comm-ring-1 set) *plus (*) 0 1*
 <proof>

At an integral domain, the locale *integral-domain* holds: the type class provides nontriviality ($0 \neq 1$) and the absence of zero divisors ($((?a * ?b = 0) = (?a = 0 \vee ?b = 0))$).

lemma *idom-TC*: *integral-domain* (*UNIV* :: 'a :: idom set) *plus (*) 0 1*
 <proof>

interpretation *field-TC*: *Field* *UNIV* :: 'a :: field set *plus (*) 0 1*
 <proof>

The interpretation gives, for every type-class field, the locale-based field structure on the whole type; e.g. its multiplicative inverse coincides with *inverse* on nonzero elements.

lemma *TC-inverse-eq*:
 fixes *a* :: 'a :: field
 assumes $a \neq 0$
 shows *field-TC.multiplicative.inverse* *a* = *inverse* *a*
 <proof>

end

31 Generic field automorphisms and fixed fields

theory *Galois-Automorphism*
 imports *Group-Theory Subfield*
 begin

This theory supplies the representation-independent core of the Galois development. A field automorphism is a bijective endomorphism of the ambient type whose field laws hold on a subfield *K* and which fixes a base subset *F*. The definition therefore works for every type-class field; the complex-specific locale and conjugation lemmas remain in the complex-field extension theory.

31.1 Automorphisms are additive, multiplicative and fix the fixed field

definition *field-auto* ::
 'a :: field set $\Rightarrow$ 'a set $\Rightarrow$ ('a $\Rightarrow$ 'a) set **where**
field-auto *K F* =
 $\{\sigma. \sigma \in K \rightarrow_E K \wedge \text{bij-betw } \sigma K K$
 $\wedge (\forall x \in K. \forall y \in K. \sigma (x + y) = \sigma x + \sigma y)$

$$\begin{aligned} & \wedge (\forall x \in K. \forall y \in K. \sigma (x * y) = \sigma x * \sigma y) \\ & \wedge \sigma 1 = 1 \\ & \wedge (\forall x \in F. \sigma x = x) \} \end{aligned}$$

lemma *field-auto-mem-iff*:

$$\begin{aligned} & \sigma \in \text{field-auto } K \ F \longleftrightarrow \\ & \sigma \in K \rightarrow_E K \wedge \text{bij-betw } \sigma \ K \ K \\ & \wedge (\forall x \in K. \forall y \in K. \sigma (x + y) = \sigma x + \sigma y) \\ & \wedge (\forall x \in K. \forall y \in K. \sigma (x * y) = \sigma x * \sigma y) \\ & \wedge \sigma 1 = 1 \\ & \wedge (\forall x \in F. \sigma x = x) \\ & \langle \text{proof} \rangle \end{aligned}$$

lemma *field-auto-zero*:

$$\begin{aligned} & \text{assumes } K\text{sub: Subfield } K \text{ and } s: \sigma \in \text{field-auto } K \ F \\ & \text{shows } \sigma 0 = 0 \\ & \langle \text{proof} \rangle \end{aligned}$$

lemma *field-auto-one*:

$$\begin{aligned} & \sigma \in \text{field-auto } K \ F \implies \sigma 1 = 1 \\ & \langle \text{proof} \rangle \end{aligned}$$

lemma *field-auto-closed*:

$$\begin{aligned} & \text{assumes } \sigma \in \text{field-auto } K \ F \text{ and } x \in K \\ & \text{shows } \sigma x \in K \\ & \langle \text{proof} \rangle \end{aligned}$$

lemma *field-auto-add*:

$$\begin{aligned} & \text{assumes } s: \sigma \in \text{field-auto } K \ F \text{ and } x: x \in K \text{ and } y: y \in K \\ & \text{shows } \sigma (x + y) = \sigma x + \sigma y \\ & \langle \text{proof} \rangle \end{aligned}$$

lemma *field-auto-mult*:

$$\begin{aligned} & \text{assumes } s: \sigma \in \text{field-auto } K \ F \text{ and } x: x \in K \text{ and } y: y \in K \\ & \text{shows } \sigma (x * y) = \sigma x * \sigma y \\ & \langle \text{proof} \rangle \end{aligned}$$

lemma *field-auto-uminus*:

$$\begin{aligned} & \text{assumes } K\text{sub: Subfield } K \ \sigma \in \text{field-auto } K \ F \text{ and } x: x \in K \\ & \text{shows } \sigma (-x) = -\sigma x \\ & \langle \text{proof} \rangle \end{aligned}$$

Additivity extends to finite sums, by induction on the index set.

lemma *field-auto-sum*:

$$\begin{aligned} & \text{assumes } K: \text{Subfield } K \text{ and } s: \sigma \in \text{field-auto } K \ F \\ & \text{and } x: \bigwedge j. j \in J \implies x j \in K \\ & \text{shows } \sigma (\sum j \in J. x j) = (\sum j \in J. \sigma (x j)) \\ & \langle \text{proof} \rangle \end{aligned}$$

lemma *field-auto-as-hom*:

assumes *Ksub*: Subfield K **and** $s: \sigma \in \text{field-auto } K \ F$
shows *field-hom-on* $K \ \sigma$
 $\langle \text{proof} \rangle$

lemma *field-auto-inverse-on*:
assumes *Ksub*: Subfield K **and** $s: \sigma \in \text{field-auto } K \ F$ **and** $x: x \in K$
shows $\sigma (\text{inverse } x) = \text{inverse } (\sigma \ x)$
 $\langle \text{proof} \rangle$

lemma *field-auto-subset-Sym*:
 $\text{field-auto } K \ F \subseteq \text{transformations.Sym } K$
 $\langle \text{proof} \rangle$

lemma *identity-apply*: $x \in S \implies \text{identity } S \ x = x$
 $\langle \text{proof} \rangle$

lemma *field-auto-identity*:
assumes *Ksub*: Subfield K **and** *FK*: $F \subseteq K$
shows *identity* $K \in \text{field-auto } K \ F$
 $\langle \text{proof} \rangle$

lemma *field-auto-compose*:
assumes *Ksub*: Subfield K **and** *FK*: $F \subseteq K$
and $s: \sigma \in \text{field-auto } K \ F$ **and** $t: \tau \in \text{field-auto } K \ F$
shows *compose* $K \ \sigma \ \tau \in \text{field-auto } K \ F$
 $\langle \text{proof} \rangle$

lemma *field-auto-inverse*:
assumes *Ksub*: Subfield K **and** $s: \sigma \in \text{field-auto } K \ F$ **and** *FK*: $F \subseteq K$
shows *restrict* $(\text{inv-into } K \ \sigma) \ K \in \text{field-auto } K \ F$
 $\langle \text{proof} \rangle$

context
fixes $K \ F :: 'a :: \text{field set}$
assumes *Ksub*: Subfield K **and** *FK*: $F \subseteq K$
begin

interpretation *T*: *transformations* $K \ \langle \text{proof} \rangle$

lemma *field-auto-restrict-inv-in-Sym*:
assumes $a \in T.\text{Sym}$
shows *restrict* $(\text{inv-into } K \ a) \ K \in T.\text{Sym}$
 $\langle \text{proof} \rangle$

lemma *field-auto-Sym-inverse-eq-inv-into*:
assumes $a \in T.\text{Sym}$
shows $T.\text{symmetric.inverse } a = \text{restrict } (\text{inv-into } K \ a) \ K$
 $\langle \text{proof} \rangle$

theorem *field-auto-subgroup*:

Subgroup (*field-auto* $K\ F$) (*transformations.Sym* K) (*compose* K) (*identity* K)
 ⟨*proof*⟩

corollary *field-auto-group*:
Group (*field-auto* $K\ F$) (*compose* K) (*identity* K)
 ⟨*proof*⟩

end

definition *fixed-field* ::
 'a :: field set $\Rightarrow$ ('a $\Rightarrow$ 'a) set $\Rightarrow$ 'a set **where**
fixed-field $K\ H \equiv \{x \in K. \forall \sigma \in H. \sigma\ x = x\}$

lemma *fixed-field-subset*: *fixed-field* $K\ H \subseteq K$
 ⟨*proof*⟩

lemma *fixed-field-memI*:
 $\llbracket x \in K; \bigwedge \sigma. \sigma \in H \Longrightarrow \sigma\ x = x \rrbracket \Longrightarrow x \in \text{fixed-field } K\ H$
 ⟨*proof*⟩

lemma *fixed-field-memD*:
 $x \in \text{fixed-field } K\ H \Longrightarrow x \in K \wedge (\forall \sigma \in H. \sigma\ x = x)$
 ⟨*proof*⟩

lemma *fixed-field-antitone*:
 $H \subseteq H' \Longrightarrow \text{fixed-field } K\ H' \subseteq \text{fixed-field } K\ H$
 ⟨*proof*⟩

lemma *fixed-field-empty*: *fixed-field* $K\ \{\} = K$
 ⟨*proof*⟩

lemma *field-auto-fixed-set-subfield*:
assumes *Ksub*: *Subfield* K **and** *s*: $\sigma \in \text{field-auto } K\ F$
shows *Subfield* $\{x \in K. \sigma\ x = x\}$ (**is** *Subfield* ? KK)
 ⟨*proof*⟩

lemma *field-auto-fixed-field-subfield*:
assumes *Ksub*: *Subfield* K **and** *H*: $H \subseteq \text{field-auto } K\ F$
shows *Subfield* (*fixed-field* $K\ H$)
 ⟨*proof*⟩

lemma *field-auto-base-subset-fixed-field*:
assumes *H*: $H \subseteq \text{field-auto } K\ F$ **and** *FK*: $F \subseteq K$
shows $F \subseteq \text{fixed-field } K\ H$
 ⟨*proof*⟩

lemma *field-auto-faithful*:
assumes *gen*: $K = \text{generate-field } (F \cup X)$
and *s*: $\sigma \in \text{field-auto } K\ F$
and *fixX*: $\bigwedge x. x \in X \Longrightarrow \sigma\ x = x$

```

    and  $x \in K$ 
  shows  $\sigma x = x$ 
  <proof>
end

```

32 Subfields of the Complex Numbers and their Automorphisms

```

theory Complex-Field-Extension
  imports Ring-Theory Subfield Galois-Automorphism
    HOL-Computational-Algebra.Fundamental-Theorem-Algebra
begin

```

Working inside the algebraically closed field *complex*, we develop just enough field theory for the Galois-theoretic study of polynomials over $\mathbb{Q}$:

- a *subfield of $\mathbb{C}$* as a subset closed under the field operations, which is then a model of the locale-based *Field* (with the type-class operations);
- the *field automorphisms* of such a subfield fixing a base subfield, which form a subgroup of the symmetric group on the carrier — the Galois group;
- complex conjugation as such an automorphism.

This bridges the locale algebra to Isabelle’s type-class fields, and provides the group on which *Group-Action* and the symmetric-group results operate when the Galois group is realised on the (finite) set of roots.

Ring-Theory suppresses the HOL arithmetic notation; restore it locally.

```

notation plus (infixl <+> 65)
notation minus (infixl <-> 65)
unbundle uminus-syntax

```

32.1 Subfields of $\mathbb{C}$

A subfield of *complex*: a subset containing *0* and *1* and closed under addition, negation, multiplication and inversion of nonzero elements.

```

locale complex-subfield =
  fixes  $K :: \text{complex set}$ 
  assumes zero-in:  $0 \in K$ 
    and one-in:  $1 \in K$ 
    and add-closed:  $\llbracket x \in K; y \in K \rrbracket \implies x + y \in K$ 
    and minus-closed:  $x \in K \implies -x \in K$ 
    and mult-closed:  $\llbracket x \in K; y \in K \rrbracket \implies x * y \in K$ 
    and inverse-closed:  $\llbracket x \in K; x \neq 0 \rrbracket \implies \text{inverse } x \in K$ 

```

begin

lemma *diff-closed*: $\llbracket x \in K; y \in K \rrbracket \implies x - y \in K$
<proof>

Built bottom-up, exactly as for the *GF2* and *Ring-Theory* witnesses.

lemma *add-group*: *Group* K $(+)$ 0
<proof>

lemma *add-abelian*: *Abelian-Group* K $(+)$ 0
<proof>

lemma *mult-cmonoid*: *commutative-monoid* K $(*)$ 1
<proof>

lemma *is-ring*: *Ring* K $(+)$ $(*)$ 0 1
<proof>

lemma *is-commutative-ring*: *commutative-ring* K $(+)$ $(*)$ 0 1
<proof>

Every subfield of *complex* is a locale-based *Field*.

lemma *is-field*: *Field* K $(+)$ $(*)$ 0 1
<proof>

end

The rationals and the reals are subfields of $\mathbb{C}$.

lemma *complex-subfield-Reals*: *complex-subfield* $\mathbb{R}$
<proof>

lemma *complex-subfield-Rats*: *complex-subfield* $\mathbb{Q}$
<proof>

corollary *field-Rats*: *Field* $\mathbb{Q}$ $(+)$ $(*)$ $(0::\text{complex})$ 1
<proof>

corollary *field-Reals*: *Field* $\mathbb{R}$ $(+)$ $(*)$ $(0::\text{complex})$ 1
<proof>

32.2 Field automorphisms and the Galois group

The identity is an automorphism.

lemma *identity-field-auto*:
 assumes *Ksub*: *complex-subfield* K **and** *FK*: $F \subseteq K$
 shows *identity* $K \in \text{field-auto } K F$
<proof>

Automorphisms are closed under composition.

lemma *compose-field-auto*:
assumes *Ksub*: *complex-subfield* *K* **and** *FK*: $F \subseteq K$
and *s*: $\sigma \in \text{field-auto } K \ F$ **and** *t*: $\tau \in \text{field-auto } K \ F$
shows *compose* *K* $\sigma \ \tau \in \text{field-auto } K \ F$
 $\langle \text{proof} \rangle$

The inverse of an automorphism (extended by the identity outside K) is an automorphism: it is a ring homomorphism because σ is injective on K .

lemma *field-auto-inv*:
assumes *Ksub*: *complex-subfield* *K* **and** *s*: $\sigma \in \text{field-auto } K \ F$ **and** *FK*: $F \subseteq K$
shows *restrict* (*inv-into* *K* σ) *K* $\in \text{field-auto } K \ F$
 $\langle \text{proof} \rangle$

Helper lemmas relating the symmetric-group inverse to the functional inverse.

lemma *cnj-invol-bij*: $(\bigwedge x. x \in K \implies \text{cnj } x \in K) \implies \text{bij-betw } (\text{restrict } \text{cnj } K) \ K \ K$
 $\langle \text{proof} \rangle$

context
fixes *K F* :: *complex set*
assumes *Ksub*: *complex-subfield* *K* **and** *FK*: $F \subseteq K$
begin

interpretation *T*: *transformations* *K* $\langle \text{proof} \rangle$

lemma *restrict-inv-in-Sym*:
assumes *a*: $a \in T.\text{Sym}$
shows *restrict* (*inv-into* *K* *a*) *K* $\in T.\text{Sym}$
 $\langle \text{proof} \rangle$

lemma *Sym-inverse-eq-inv-into*:
assumes *a* $\in T.\text{Sym}$
shows *T.symmetric.inverse* *a* = *restrict* (*inv-into* *K* *a*) *K*
 $\langle \text{proof} \rangle$

The Galois group is a subgroup of the symmetric group on K .

theorem *Galois-group-subgroup*:
Subgroup (*field-auto* *K F*) (*transformations.Sym* *K*) (*compose* *K*) (*identity* *K*)
 $\langle \text{proof} \rangle$

Hence the Galois group is a group.

corollary *Galois-group-Group*:
Group (*field-auto* *K F*) (*compose* *K*) (*identity* *K*)
 $\langle \text{proof} \rangle$

end

32.3 Complex conjugation as an automorphism

If K is closed under conjugation and the base F is real, then complex conjugation (restricted to K) is an element of the Galois group. This is the automorphism that, for a polynomial with real coefficients, transposes a pair of complex conjugate roots.

lemma *cnj-field-auto*:

assumes K_{sub} : *complex-subfield* K **and** $\text{cnj}K$: $\text{cnj} \restriction K = K$ **and** FR : $F \subseteq \mathbb{R}$ **and** FK : $F \subseteq K$

shows *restrict cnj* $K \in \text{field-auto } K F$

<proof>

32.4 The subfield generated by a set; splitting subfields

The whole of *complex* and the intersection of a non-empty family of subfields are subfields; hence the *subfield generated by* a set of complex numbers is well defined as the intersection of all subfields containing it.

lemma *complex-subfield-UNIV*: *complex-subfield* ($UNIV :: \text{complex set}$)

<proof>

lemma *complex-subfield-Inter*:

assumes $C \neq \{\}$ **and** $[simp]$: $\bigwedge K. K \in C \implies \text{complex-subfield } K$

shows *complex-subfield* $(\bigcap C)$

<proof>

The subfield generated by X : the smallest subfield containing X .

definition *gen-subfield* :: *complex set* $\Rightarrow$ *complex set*

where *gen-subfield* $X = \bigcap \{K. \text{complex-subfield } K \wedge X \subseteq K\}$

lemma *gen-subfield-is-subfield*: *complex-subfield* (*gen-subfield* X)

<proof>

lemma *gen-subfield-subset*: $X \subseteq \text{gen-subfield } X$

<proof>

lemma *gen-subfield-minimal*:

assumes *complex-subfield* K $X \subseteq K$

shows *gen-subfield* $X \subseteq K$

<proof>

The complex-specific and generic generated subfields are the same. Keeping this bridge next to *gen-subfield* makes it available to both the Galois and rational developments.

lemma *gen-subfield-eq-generate-field*: *gen-subfield* $X = \text{generate-field } X$

<proof>

For the splitting subfield $\mathbb{Q}(X)$ we generate from $\mathbb{Q} \cup X$: it contains the rationals and the elements of X .

lemma *Rats-subset-gen-subfield*: $\mathbb{Q} \subseteq \text{gen-subfield } (\mathbb{Q} \cup X)$
 $\langle \text{proof} \rangle$

lemma *roots-subset-gen-subfield*: $X \subseteq \text{gen-subfield } (\mathbb{Q} \cup X)$
 $\langle \text{proof} \rangle$

Conjugation is a field automorphism of *complex*, so the image of a subfield under conjugation is again a subfield.

lemma *cnj-image-eq*: $\text{cnj } ' K = \{y. \text{ cnj } y \in K\}$
 $\langle \text{proof} \rangle$

lemma *complex-subfield-cnj-image*:
assumes *complex-subfield* K
shows *complex-subfield* $(\text{cnj } ' K)$
 $\langle \text{proof} \rangle$

A subfield generated by a conjugation-closed set is itself conjugation-closed.

lemma *gen-subfield-cnj-closed*:
assumes $\text{cnj } ' Y = Y$
shows $\text{cnj } ' \text{gen-subfield } Y = \text{gen-subfield } Y$
 $\langle \text{proof} \rangle$

Conjugation fixes the rationals.

lemma *cnj-of-rat*: $\text{cnj } (\text{of-rat } q) = \text{of-rat } q$
 $\langle \text{proof} \rangle$

lemma *cnj-fixes-Rats*: $x \in \mathbb{Q} \implies \text{cnj } x = x$
 $\langle \text{proof} \rangle$

lemma *cnj-Rats-eq*: $\text{cnj } ' \mathbb{Q} = \mathbb{Q}$
 $\langle \text{proof} \rangle$

Hence the splitting subfield $\mathbb{Q}(X)$ of a conjugation-closed set X (e.g. the set of roots of a real-coefficient polynomial) is conjugation-closed — so by *cnj-field-auto* complex conjugation restricts to an element of its Galois group over $\mathbb{Q}$.

lemma *splitting-subfield-cnj-closed*:
assumes $\text{cnj } ' X = X$
shows $\text{cnj } ' \text{gen-subfield } (\mathbb{Q} \cup X) = \text{gen-subfield } (\mathbb{Q} \cup X)$
 $\langle \text{proof} \rangle$

32.5 Faithfulness of the action on a generating set

The set of points fixed by an automorphism is itself a subfield (it contains 0 and 1, and is closed under the field operations, since the automorphism respects them).

lemma *fixed-set-subfield*:
assumes *Ksub*: *complex-subfield* K **and** $s: \sigma \in \text{field-auto } K \ F$

shows *complex-subfield* $\{x \in K. \sigma x = x\}$
 <proof>

An automorphism of $K = F(X)$ fixing the base F and every generator in X is the identity on K : the fixed subfield contains $F \cup X$, hence all of the generated field K . Thus the Galois group acts *faithfully* on any generating set — in particular on the roots of a polynomial whose splitting field is K .

theorem *field-auto-faithful*:

assumes *Ksub*: *complex-subfield* K
and *gen*: $K = \text{gen-subfield } (F \cup X)$
and *s*: $\sigma \in \text{field-auto } K F$
and *fixX*: $\bigwedge x. x \in X \implies \sigma x = x$
shows $\forall x \in K. \sigma x = x$
 <proof>

Specialised to the splitting field: two automorphisms agreeing on the generators agree everywhere, so the permutation action on the generators is faithful.

corollary *galois-faithful-on-generators*:

assumes *s*: $\sigma \in \text{field-auto } (\text{gen-subfield } (F \cup X)) F$
and *fixX*: $\bigwedge x. x \in X \implies \sigma x = x$
shows $\forall x \in \text{gen-subfield } (F \cup X). \sigma x = x$
 <proof>

end

33 Iterating the Iso-Extension over a Finite Set of Algebraic Elements

theory *Iso-Extension-Tower*

imports *Iso-Extension Complex-Field-Extension*

begin

Starting from a homomorphism f defined on a subfield K , we extend it across a whole finite family of elements that are algebraic over K , reaching a homomorphism on the generated subfield *generate-field* $(K \cup A)$. Each extension step uses the single-element *field-hom-on.iso-extension*; the only external requirement is that the target field 'b contains a root for every non-constant polynomial, i.e. is “closed enough”. Over *complex* this is the fundamental theorem of algebra.

The minimal polynomial is monic, so its image under f keeps the same degree.

lemma (in *field-hom-on*) *map-poly-minpoly-degree*:

assumes *m*: *is-minpoly* $K a m$
shows *degree* $(\text{map-poly } f m) = \text{degree } m$
 <proof>

A subfield is generated by itself.

lemma (in *Subfield*) *generate-field-self*: *generate-field* $K = K$
 ⟨*proof*⟩

The single-element extension, packaged with the closed-target hypothesis discharging the existence of a root of the mapped minimal polynomial.

lemma *field-hom-on-extend-one*:
 assumes *f*: *field-hom-on* K *f*
 and *alg*: *algebraic-over* K *a*
 and *closed*: $\bigwedge q :: 'b :: \text{field poly. } 0 < \text{degree } q \implies \exists b. \text{poly } q \ b = 0$
 shows $\exists g. \text{field-hom-on } (\text{generate-field } (K \cup \{a\})) \ g \wedge (\forall x \in K. \ g \ x = (f :: 'a :: \text{field} \Rightarrow 'b) \ x)$
 ⟨*proof*⟩

Iterating over a finite set: *f* extends to the subfield generated by K together with finitely many elements algebraic over K .

theorem *field-hom-on-extend-finite*:
 assumes *closed*: $\bigwedge q :: 'b :: \text{field poly. } 0 < \text{degree } q \implies \exists b. \text{poly } q \ b = 0$
 and *finite* A
 and *field-hom-on* K ($f :: 'a :: \text{field} \Rightarrow 'b$)
 and $\bigwedge a. a \in A \implies \text{algebraic-over } K \ a$
 shows $\exists g. \text{field-hom-on } (\text{generate-field } (K \cup A)) \ g \wedge (\forall x \in K. \ g \ x = f \ x)$
 ⟨*proof*⟩

The generic field-extension development works over an arbitrary ambient $'a$ via the *Subfield* and *field-hom-on* locales. The complex-specific Galois theory of *Complex-Field-Extension* instead fixes *complex* and packages automorphisms as the set *field-auto*. Now we connect the two.

The two subfield notions coincide on *complex*.

lemma *complex-subfield-iff-subfield*: *complex-subfield* $K \longleftrightarrow \text{Subfield } K$
 ⟨*proof*⟩

A field homomorphism on K (in the generic sense) that maps K *onto* itself and fixes a base subfield F pointwise restricts to an element of the complex Galois group *field-auto* $K \ F$.

lemma *field-hom-on-imp-field-auto*:
 assumes *hom*: *field-hom-on* K σ
 and *onto*: $\sigma \text{ ' } K = K$
 and *fix-F*: $\bigwedge x. x \in F \implies \sigma \ x = x$ and *FK*: $F \subseteq K$
 shows *restrict* $\sigma \ K \in \text{field-auto } K \ F$
 ⟨*proof*⟩

end

34 Transitivity of the Galois Action on the Roots

theory *Galois-Transitivity*

imports *Iso-Extension-Tower*
begin

The capstone of the iso-extension development: for a polynomial Phi over a base subfield F whose root set R is finite and consists of elements algebraic over F , any two roots related by the minimal polynomial of one of them are connected by an F -fixing automorphism of the splitting field $K = \text{generate-field } (F \cup R)$. This is *transitivity* of the Galois action on the roots, the key input to identifying the Galois group of an irreducible quintic with S_5 .

The construction has three parts: a base isomorphism $F(a) \cong F(b)$ sending a to b (the single-step *field-hom-on.iso-extension*); its extension across the remaining roots (*field-hom-on-extend-finite*); and the observation that the resulting embedding permutes the finite root set and therefore maps the splitting field onto itself (*field-hom-on.image-generate-field-eq*), i.e. is an automorphism.

The identity is a homomorphism of any subfield — the starting point of the tower.

lemma *field-hom-on-id*:

assumes *Subfield K* **shows** *field-hom-on K* $(\lambda x. x)$

<proof>

An endomorphism fixing F pointwise leaves every polynomial over F unchanged under *map-poly*.

lemma *map-poly-fix*:

fixes $f :: 'a :: \text{field} \Rightarrow 'a$

assumes *hom: field-hom-on K f* **and** *sfF: Subfield F*

and $p: p \in \text{poly-over } F$ **and** *fixF: $\bigwedge x. x \in F \implies f x = x$*

shows *map-poly f p = p*

<proof>

Hence such an endomorphism sends a root of Phi to a root of Phi .

lemma *hom-preserves-roots*:

fixes $f :: 'a :: \text{field} \Rightarrow 'a$

assumes *hom: field-hom-on K f* **and** *sfF: Subfield F* **and** *FK: $F \subseteq K$*

and *fixF: $\bigwedge x. x \in F \implies f x = x$*

and *Phi: $\text{Phi} \in \text{poly-over } F$* **and** $r: r \in K$ **and** *root: $\text{poly } \text{Phi } r = 0$*

shows *poly Phi (f r) = 0*

<proof>

theorem *galois-transitive*:

fixes $F :: 'a :: \text{field set}$

assumes *sfF: Subfield F*

and *finR: finite R*

and *algR: $\bigwedge r. r \in R \implies \text{algebraic-over } F r$*

and *Phi: $\text{Phi} \in \text{poly-over } F$*

and *rootsR: $R = \{r. \text{poly } \text{Phi } r = 0\}$*

and *closed: $\bigwedge q :: 'a \text{ poly. } 0 < \text{degree } q \implies \exists b. \text{poly } q b = 0$*

and $a: a \in R$ **and** $b: b \in R$

and m : *is-minpoly* F a m **and** mb : *poly* m $b = 0$
defines $K \equiv \text{generate-field } (F \cup R)$
shows $\exists \sigma. \text{field-hom-on } K \ \sigma \wedge \sigma \ a = b \wedge (\forall x \in F. \sigma \ x = x) \wedge \sigma \restriction K = K$
 $\langle \text{proof} \rangle$

For an *irreducible* polynomial the minimal-polynomial hypothesis is automatic, so the Galois action is transitive on its roots: any two roots are related by an F -fixing automorphism of the splitting field.

corollary *galois-transitive-irreducible*:
fixes $F :: 'a :: \text{field set}$
assumes $\text{sf}F$: *Subfield* F
and $\text{fin}R$: *finite* R
and $\text{alg}R$: $\bigwedge r. r \in R \implies \text{algebraic-over } F \ r$
and Phi : $\text{Phi} \in \text{poly-over } F$
and irr : *irreducible-over* $F \ \text{Phi}$
and $\text{roots}R$: $R = \{r. \text{poly } \text{Phi} \ r = 0\}$
and closed : $\bigwedge q :: 'a \text{ poly}. 0 < \text{degree } q \implies \exists b. \text{poly } q \ b = 0$
and a : $a \in R$ **and** b : $b \in R$
defines $K \equiv \text{generate-field } (F \cup R)$
shows $\exists \sigma. \text{field-hom-on } K \ \sigma \wedge \sigma \ a = b \wedge (\forall x \in F. \sigma \ x = x) \wedge \sigma \restriction K = K$
 $\langle \text{proof} \rangle$

34.1 Rational Subfields and Irreducibility over the Rationals

A complex polynomial all of whose coefficients are rational is the *of-rat*-image of a rational polynomial.

lemma *poly-over-Rats-imp-map-of-rat*:
assumes $p \in \text{poly-over } (\mathbb{Q} :: \text{complex set})$
shows $\exists r :: \text{rat poly}. p = \text{map-poly of-rat } r$
 $\langle \text{proof} \rangle$

lemma *map-poly-of-rat-mult*:
 $\text{map-poly of-rat } (a * b) = (\text{map-poly of-rat } a :: \text{complex poly}) * \text{map-poly of-rat } b$
 $\langle \text{proof} \rangle$

lemma *map-poly-of-rat-inj*: $\text{inj } (\text{map-poly of-rat} :: \text{rat poly} \Rightarrow \text{complex poly})$
 $\langle \text{proof} \rangle$

lemma *degree-map-poly-of-rat*:
 $\text{degree } (\text{map-poly of-rat } r :: \text{complex poly}) = \text{degree } r$
 $\langle \text{proof} \rangle$

Irreducibility of a rational polynomial transfers to irreducibility *over* $\mathbb{Q}$ of its complex image: a factorisation over $\mathbb{Q}$ pulls back along the injective ring homomorphism *map-poly of-rat* to a factorisation of the rational polynomial.

lemma *irreducible-imp-irreducible-over-Rats*:
fixes $r :: \text{rat poly}$

assumes *irr*: irreducible *r* **and** *deg*: degree *r* > 0
shows irreducible-over ($\mathbb{Q} :: \text{complex set}$) (map-poly of-rat *r*)
 <proof>

end

35 The Galois Group Acts on the Roots

theory *Galois-Action*

imports *Galois-Transitivity Symmetric-Generation Sym-Not-Solvable*
begin

A field automorphism (an element of *field-auto*) is, in particular, a field homomorphism on its subfield domain.

lemma *field-auto-imp-field-hom-on*:

assumes *K*: Subfield *K* **and** *s*: $\sigma \in \text{field-auto } K \ F$
shows *field-hom-on* *K* σ
 <proof>

An automorphism fixing *F* maps a root in *K* of an *F*-polynomial to a root: it permutes the root set.

lemma *field-auto-maps-root*:

assumes *K*: Subfield *K* **and** *sfF*: Subfield *F* **and** *FK*: $F \subseteq K$
and *s*: $\sigma \in \text{field-auto } K \ F$ **and** *Phi*: $\text{Phi} \in \text{poly-over } F$
and *r*: $r \in K$ **and** *root*: $\text{poly } \text{Phi } r = 0$
shows $\text{poly } \text{Phi } (\sigma \ r) = 0$
 <proof>

The Galois group acts on the root set by restriction. We package this as a *Group-Action* of *field-auto* *K F* on the finite root set *X*.

context

fixes *K F* :: 'a :: field set **and** *Phi* :: 'a poly **and** *X* :: 'a set
assumes *K*: Subfield *K* **and** *sfF*: Subfield *F* **and** *FK*: $F \subseteq K$
and *Phi*: $\text{Phi} \in \text{poly-over } F$
and *Xdef*: $X = \{r. \text{poly } \text{Phi } r = 0\}$ **and** *XK*: $X \subseteq K$ **and** *finX*: finite *X*

begin

lemma *field-auto-image-root-subset*:

assumes *s*: $\sigma \in \text{field-auto } K \ F$
shows $\sigma \ ` X \subseteq X$
 <proof>

lemma *field-auto-bij-on-root*:

assumes *s*: $\sigma \in \text{field-auto } K \ F$
shows *bij-betw* $\sigma \ X \ X$
 <proof>

lemma *ga-sym*:

assumes $s: \sigma \in \text{field-auto } K \ F$
shows $\text{restrict } \sigma \ X \in \text{transformations.Sym } X$
 $\langle \text{proof} \rangle$

lemma *ga-compose*:

assumes $s: \sigma \in \text{field-auto } K \ F$ **and** $t: \tau \in \text{field-auto } K \ F$
shows $\text{restrict } (\text{compose } K \ \sigma \ \tau) \ X = \text{compose } X \ (\text{restrict } \sigma \ X) \ (\text{restrict } \tau \ X)$
 $\langle \text{proof} \rangle$

lemma *ga-unit*: $\text{restrict } (\text{identity } K) \ X = \text{identity } X$
 $\langle \text{proof} \rangle$

lemma *galois-group-action*:

Group-Action $(\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K) \ (\lambda \sigma. \text{restrict } \sigma \ X) \ X$
 $\langle \text{proof} \rangle$

When the ambient field is generated by the base and the root set, the root action is faithful: an automorphism fixing every root fixes the generators and is therefore the identity.

lemma *galois-action-faithful*:

assumes $K_{\text{gen}}: K = \text{generate-field } (F \cup X)$
shows $\forall \sigma \in \text{field-auto } K \ F.$
 $(\forall x \in X. \text{restrict } \sigma \ X \ x = x) \longrightarrow \sigma = \text{identity } K$
 $\langle \text{proof} \rangle$

lemma *finite-galois-group*:

assumes $K_{\text{gen}}: K = \text{generate-field } (F \cup X)$
shows $\text{finite } (\text{field-auto } K \ F)$
 $\langle \text{proof} \rangle$

end

Compatibility wrapper for the historical complex-only spelling. The generic theorem uses *generate-field*; *gen-subfield* is the same construction on *complex* and remains available to existing clients.

lemma *finite-field-auto*:

fixes $K \ F :: \text{complex set}$ **and** $\Phi :: \text{complex poly}$ **and** $X :: \text{complex set}$
assumes $K: \text{Subfield } K$ **and** $\text{sfF}: \text{Subfield } F$ **and** $\text{FK}: F \subseteq K$
and $\Phi: \Phi \in \text{poly-over } F$
and $X_{\text{def}}: X = \{r. \text{poly } \Phi \ r = 0\}$ **and** $XK: X \subseteq K$ **and** $\text{finX}: \text{finite } X$
and $K_{\text{gen}}: K = \text{gen-subfield } (F \cup X)$
shows $\text{finite } (\text{field-auto } K \ F)$
 $\langle \text{proof} \rangle$

Transitivity of the Galois action on the roots of an irreducible polynomial: any root is carried to any other by some F -automorphism (via *galois-transitive-irreducible* and the bridge *field-hom-on-imp-field-auto*).

lemma *galois-action-transitive*:

```

fixes  $F :: 'a :: \text{field set}$  and  $\text{Phi} :: 'a \text{ poly}$  and  $X :: 'a \text{ set}$ 
assumes  $\text{sfF}: \text{Subfield } F$ 
and  $\text{finX}: \text{finite } X$ 
and  $\text{Phi}: \text{Phi} \in \text{poly-over } F$ 
and  $\text{irr}: \text{irreducible-over } F \text{ Phi}$ 
and  $\text{Xdef}: X = \{r. \text{poly Phi } r = 0\}$ 
and  $\text{closed}: \bigwedge q :: 'a \text{ poly}. 0 < \text{degree } q \implies \exists b. \text{poly } q \text{ } b = 0$ 
and  $a: a \in X$  and  $b: b \in X$ 
defines  $K \equiv \text{generate-field } (F \cup X)$ 
shows  $\exists \sigma \in \text{field-auto } K \text{ } F. \text{ restrict } \sigma \text{ } X \text{ } a = b$ 
 $\langle \text{proof} \rangle$ 

end

```

36 Embedding a Faithful Finite Action into a Concrete Symmetric Group

```

theory Action-To-Sn
imports Symmetric-Group Group-Operations
begin

```

36.1 Relabelling Symmetric Groups along a Bijection

Relabelling along a bijection e from $\{0..<n\}$ to a finite set X transports the symmetric group on X to S_n .

```

definition  $\text{pull} :: \text{nat} \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow \text{nat})$ 
where  $\text{pull } n \text{ } e \text{ } a = (\lambda i. \text{ if } i < n \text{ then } \text{inv-into } \{0..<n\} \text{ } e \text{ } (a \text{ } (e \text{ } i)) \text{ else } i)$ 

```

```

context
fixes  $n :: \text{nat}$  and  $e :: \text{nat} \Rightarrow 'a$  and  $X :: 'a \text{ set}$ 
assumes  $\text{ebij}: \text{bij-betw } e \text{ } \{0..<n\} \text{ } X$ 
begin

```

```

lemma  $e\text{-in-}X: i < n \implies e \text{ } i \in X$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma  $\text{inv-}e\text{-lt}: x \in X \implies \text{inv-into } \{0..<n\} \text{ } e \text{ } x < n$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma  $\text{inv-}e\text{-}e: i < n \implies \text{inv-into } \{0..<n\} \text{ } e \text{ } (e \text{ } i) = i$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma  $e\text{-inv-}e: x \in X \implies e \text{ } (\text{inv-into } \{0..<n\} \text{ } e \text{ } x) = x$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma  $\text{pull-permutes}$ :
assumes  $a: a \in \text{transformations.Sym } X$ 
shows  $\text{pull } n \text{ } e \text{ } a \text{ permutes } \{0..<n\}$ 

```


<proof>

lemma *pull-compose:*

assumes *a: a ∈ transformations.Sym X and b: b ∈ transformations.Sym X*

shows *pull n e (compose X a b) = pull n e a ∘ pull n e b*

<proof>

lemma *pull-unit: pull n e (identity X) = id*

<proof>

lemma *pull-in-Sn: a ∈ transformations.Sym X ⇒ pull n e a ∈ Sn n*

<proof>

end

context *Group-Action*

begin

Transport a faithful finite action into the concrete symmetric group S_n (card S), via an enumeration e of S .

lemma *action-pull-hom:*

assumes *ebij: bij-betw e {0.. n } S*

shows *group-homomorphism (restrict (λg. pull n e (φ g)) G) G (·) 1 (Sn n) (∘) id*

<proof>

Under faithfulness the transported homomorphism has trivial kernel, hence is injective.

lemma *action-pull-inj:*

assumes *ebij: bij-betw e {0.. n } S and faith: faithful*

shows *inj-on (restrict (λg. pull n e (φ g)) G) G*

<proof>

Hence a faithful finite action embeds G as a subgroup of S_n : the transported map is a group isomorphism onto its image.

theorem *action-embeds-Sn:*

assumes *ebij: bij-betw e {0.. n } S and faith: faithful*

shows *group-isomorphism (restrict (λg. pull n e (φ g)) G) G (·) 1*

((λg. pull n e (φ g)) ‘ G) (∘) id

<proof>

end

end

37 The Symmetric Group on the Roots and Non-Solvability

```

theory Abel-Ruffini
  imports Galois-Action Action-To-Sn
begin

```

A subgroup of S_n containing some p -cycle (conjugate to $ncycle\ p$) and some transposition is the whole group.

```

lemma subgroup-with-cycle-transpose-eq-Sn:
  fixes  $p :: nat$  and  $q :: nat \Rightarrow nat$  and  $i\ j :: nat$ 
  assumes  $p$ : prime  $p$  and  $H$ : Subgroup  $H$  ( $S_n\ p$ )  $(\circ)\ id$ 
    and  $q$ :  $q \in S_n\ p$ 
    and  $c\text{-in}$ :  $q \circ ncycle\ p \circ inv\ q \in H$ 
    and  $t\text{-in}$ : Transposition.transpose  $i\ j \in H$ 
    and  $i$ :  $i < p$  and  $jj$ :  $j < p$  and  $ij$ :  $i \neq j$ 
  shows  $H = S_n\ p$ 
  <proof>

```

The abstract group-theoretic heart of Abel–Ruffini: a finite group acting faithfully and transitively on a set of prime cardinality $(5::'a) \leq p$, whose image in $S_n\ p$ contains a transposition, is not solvable, because that image is then all of $S_n\ p$, which is not solvable.

```

theorem (in Group-Action) not-solvable-of-transitive-with-transposition:
  fixes  $e :: nat \Rightarrow 'b$  and  $p :: nat$ 
  assumes  $ebij$ : bij-betw  $e\ \{0..<p\}\ S$ 
    and  $p$ : prime  $p$  and  $p5$ :  $5 \leq p$  and  $finG$ : finite  $G$ 
    and  $faith$ : faithful
    and  $trans$ :  $\exists s \in S. orbit\ s = S$ 
    and  $transp$ :  $\exists i\ j. i < p \wedge j < p \wedge i \neq j$ 
       $\wedge Transposition.transpose\ i\ j \in (\lambda g. pull\ p\ e\ (\varphi\ g))\ 'G$ 
  shows  $\neg Group.solvable\ G\ (\cdot)\ 1$ 
  <proof>

```

end

38 Finite p -groups

```

theory P-Group
  imports Sylow-Theorems
begin

```

```

context Group
begin

```

The class equation makes the center of a nontrivial finite p -group visible arithmetically: every noncentral conjugacy orbit has cardinality divisible by p .

```

theorem p-group-prime-dvd-card-center:

```

```

fixes  $p\ n :: \text{nat}$ 
assumes  $\text{prime: prime } p$  and  $\text{fin: finite } G$  and  $\text{npos: } n > 0$ 
and  $\text{cardG: card } G = p \wedge n$ 
shows  $p \text{ dvd card center}$ 
 $\langle \text{proof} \rangle$ 

```

```

corollary  $p\text{-group-center-nontrivial:}$ 
fixes  $p\ n :: \text{nat}$ 
assumes  $\text{prime: prime } p$  and  $\text{fin: finite } G$  and  $\text{npos: } n > 0$ 
and  $\text{cardG: card } G = p \wedge n$ 
shows  $\exists z \in \text{center. } z \neq 1$ 
 $\langle \text{proof} \rangle$ 

```

A group of order p^2 is abelian. A hypothetical noncentral element has a centralizer strictly larger than the center; Lagrange then forces that centralizer to have the full group cardinality.

```

theorem  $\text{card-prime-square-imp-abelian:}$ 
fixes  $p :: \text{nat}$ 
assumes  $\text{prime: prime } p$  and  $\text{fin: finite } G$  and  $\text{cardG: card } G = p \wedge 2$ 
shows  $\text{Abelian-Group } G\ (\cdot)\ 1$ 
 $\langle \text{proof} \rangle$ 

```

end

end

39 The subring generated by a set

```

theory  $\text{Subring-Generated}$ 
imports  $\text{Ring-Theory}$ 
begin

```

Mirroring the subgroup *generate* and the ideal generated by a set, the subring generated by a subset S of a ring is the intersection of all subrings containing $S \cap R$. We first show that an intersection of a nonempty family of subrings is a subring; the whole ring is a subring of itself, so the family intersected is always nonempty.

```

context  $\text{Ring}$ 
begin

```

39.1 Intersections of subrings

The intersection of a nonempty family of subrings is a subring.

```

lemma  $\text{subring-Inter:}$ 
assumes  $\text{ne: } \mathcal{A} \neq \{\}$  and  $\text{sr: } \bigwedge S. S \in \mathcal{A} \implies \text{Subring } S\ R\ (+)\ (\cdot)\ 0\ 1$ 
shows  $\text{Subring } (\bigcap \mathcal{A})\ R\ (+)\ (\cdot)\ 0\ 1$ 
 $\langle \text{proof} \rangle$ 

```

The whole ring is a subring of itself (the top subring).

lemma *whole-subring*: $\text{Subring } R \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$
 $\langle \text{proof} \rangle$

39.2 The generated subring

The subring generated by a subset S : the intersection of all subrings containing $S \cap R$ (intersecting with R in the guard, as with subgroups and ideals, so that elements outside the carrier are ignored).

definition *subring-generated* :: 'a set $\Rightarrow$ 'a set
where *subring-generated* $S = \bigcap \{T. \text{Subring } T \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1} \wedge S \cap R \subseteq T\}$

lemma *subring-generated-is-subring*: $\text{Subring } (\text{subring-generated } S) \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$
 $\langle \text{proof} \rangle$

lemma *subring-generated-incl*: $S \cap R \subseteq \text{subring-generated } S$
 $\langle \text{proof} \rangle$

lemma *subring-generated-minimal*:
assumes $\text{Subring } T \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1} \text{ } S \cap R \subseteq T$
shows $\text{subring-generated } S \subseteq T$
 $\langle \text{proof} \rangle$

lemma *subring-generated-subset-carrier*: $\text{subring-generated } S \subseteq R$
 $\langle \text{proof} \rangle$

end

end

40 The characteristic of a ring; finite domains have prime characteristic

theory *Characteristic*
imports *Ring-Theory HOL-Computational-Algebra.Primes*
begin

The natural-number multiples $n \cdot \mathbf{1}$ of the unit form a ring homomorphism $\mathbb{N} \rightarrow R$. Its kernel determines the *characteristic* of the ring; for a finite integral domain the characteristic is a prime, and the multiples $n \cdot \mathbf{1}$ form the prime subfield. This is the first step toward the structural theorem that a finite field has prime-power order.

context *Ring*
begin

The n -fold sum $\mathbf{1} + \dots + \mathbf{1}$ (n summands), i.e. the image of n under the canonical map $\mathbb{N} \rightarrow R$.

definition *natmult* :: *nat* $\Rightarrow$ 'a
where *natmult* *n* = (((+ 1) $\smile$ *n*) 0

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

lemma *natmult-Suc* [*simp*]: *natmult* (Suc *n*) = 1 + *natmult* *n*
 $\langle$ *proof* $\rangle$

lemma *natmult-closed* [*simp*]: *natmult* *n* $\in$ *R*
 $\langle$ *proof* $\rangle$

lemma *natmult-1* [*simp*]: *natmult* 1 = 1
 $\langle$ *proof* $\rangle$

Additivity: *natmult* (*m* + *n*) = *natmult* *m* + *natmult* *n*.

lemma *natmult-add*: *natmult* (*m* + *n*) = *natmult* *m* + *natmult* *n*
 $\langle$ *proof* $\rangle$

Multiplicativity: *natmult* (*m* * *n*) = *natmult* *m* · *natmult* *n*.

lemma *natmult-mult*: *natmult* (*m* * *n*) = *natmult* *m* · *natmult* *n*
 $\langle$ *proof* $\rangle$

Cancellation across a difference: if two multiples agree, the multiple of their difference vanishes.

lemma *natmult-diff-zero*:
assumes *m* $\leq$ *n* **and** *natmult* *m* = *natmult* *n*
shows *natmult* (*n* - *m*) = 0
 $\langle$ *proof* $\rangle$

end

context *integral-domain*
begin

The *characteristic*: the least positive *n* with *n* · 1 = 0 (or 0 if no such *n* exists).

definition *characteristic* :: *nat*
where *characteristic* = (LEAST *n*. 0 < *n* $\wedge$ *natmult* *n* = 0)

In a *finite* integral domain the map *natmult* cannot be injective, so some positive multiple of 1 vanishes.

lemma *finite-natmult-not-inj*:
assumes *finite* *R* **shows** $\exists$ *n*. 0 < *n* $\wedge$ *natmult* *n* = 0
 $\langle$ *proof* $\rangle$

lemma *characteristic-pos*:
assumes *finite* *R* **shows** 0 < *characteristic* $\wedge$ *natmult* *characteristic* = 0
 $\langle$ *proof* $\rangle$

lemma *characteristic-least*:
 $\llbracket 0 < n; \text{natmult } n = \mathbf{0} \rrbracket \implies \text{characteristic} \leq n$
 $\langle \text{proof} \rangle$

The characteristic exceeds 1: $1 \cdot \mathbf{1} = \mathbf{1} \neq \mathbf{0}$ by nontriviality.

lemma *characteristic-gt-1*:
assumes *finite R* **shows** $1 < \text{characteristic}$
 $\langle \text{proof} \rangle$

The characteristic of a finite integral domain is prime. If it factored as $\text{characteristic} = a * b$ nontrivially, then $\text{natmult } a \cdot \text{natmult } b = \mathbf{0}$ would force one factor's multiple to vanish, contradicting minimality.

theorem *characteristic-prime*:
assumes *finR: finite R* **shows** *prime characteristic*
 $\langle \text{proof} \rangle$

end

end

41 Product Operator for Commutative Monoids

theory *Finite-Composite*
imports *Group-Theory*

begin

41.1 Products over Finite Sets

context *commutative-monoid* **begin**

definition *M-ify* $x \equiv \text{if } x \in M \text{ then } x \text{ else } \mathbf{1}$

definition *fincomp* $f A \equiv \text{if finite } A \text{ then } \text{Finite-Set.fold } (\lambda x y. f x \cdot M\text{-ify } y) \mathbf{1} A \text{ else } \mathbf{1}$

lemma *fincomp-empty* $[\text{simp}]$: $\text{fincomp } f \ \{\} = \mathbf{1}$
 $\langle \text{proof} \rangle$

lemma *fincomp-infinite* $[\text{simp}]$: $\text{infinite } A \implies \text{fincomp } f A = \mathbf{1}$
 $\langle \text{proof} \rangle$

lemma *left-commute*: $\llbracket a \in M; b \in M; c \in M \rrbracket \implies b \cdot (a \cdot c) = a \cdot (b \cdot c)$
 $\langle \text{proof} \rangle$

lemma *comp-fun-commute-onI*:

assumes $f \in F \rightarrow M$
shows *comp-fun-commute-on* F $(\lambda x y. f x \cdot M\text{-ify } y)$
 $\langle \text{proof} \rangle$

lemma *fincomp-closed* [*simp*]:
assumes $f \in F \rightarrow M$
shows *fincomp* $f F \in M$
 $\langle \text{proof} \rangle$

lemma *fincomp-insert* [*simp*]:
assumes F : *finite* F $a \notin F$ **and** f : $f \in F \rightarrow M$ $f a \in M$
shows *fincomp* f (*insert* $a F$) = $f a \cdot \text{fincomp } f F$
 $\langle \text{proof} \rangle$

lemma *fincomp-unit-eqI*: $(\bigwedge x. x \in A \implies f x = \mathbf{1}) \implies \text{fincomp } f A = \mathbf{1}$
 $\langle \text{proof} \rangle$

lemma *fincomp-unit* [*simp*]: *fincomp* $(\lambda i. \mathbf{1}) A = \mathbf{1}$
 $\langle \text{proof} \rangle$

lemma *funcset-Int-left* [*simp*, *intro*]:
 $\llbracket f \in A \rightarrow C; f \in B \rightarrow C \rrbracket \implies f \in A \text{ Int } B \rightarrow C$
 $\langle \text{proof} \rangle$

lemma *funcset-Un-left* [*iff*]:
 $(f \in A \text{ Un } B \rightarrow C) = (f \in A \rightarrow C \wedge f \in B \rightarrow C)$
 $\langle \text{proof} \rangle$

lemma *fincomp-Un-Int*:
 $\llbracket \text{finite } A; \text{finite } B; g \in A \rightarrow M; g \in B \rightarrow M \rrbracket \implies$
 $\text{fincomp } g (A \cup B) \cdot \text{fincomp } g (A \cap B) =$
 $\text{fincomp } g A \cdot \text{fincomp } g B$
— The reversed orientation looks more natural, but LOOPS as a simprule!
 $\langle \text{proof} \rangle$

lemma *fincomp-Un-disjoint*:
 $\llbracket \text{finite } A; \text{finite } B; A \cap B = \{\}; g \in A \rightarrow M; g \in B \rightarrow M \rrbracket$
 $\implies \text{fincomp } g (A \cup B) = \text{fincomp } g A \cdot \text{fincomp } g B$
 $\langle \text{proof} \rangle$

lemma *fincomp-comp*:
 $\llbracket f \in A \rightarrow M; g \in A \rightarrow M \rrbracket \implies \text{fincomp } (\lambda x. f x \cdot g x) A = (\text{fincomp } f A \cdot \text{fincomp } g A)$
 $\langle \text{proof} \rangle$

lemma *fincomp-cong'*:
assumes $A = B$ $g \in B \rightarrow M$ $\bigwedge i. i \in B \implies f i = g i$
shows *fincomp* $f A = \text{fincomp } g B$
 $\langle \text{proof} \rangle$

lemma *fincomp-cong*:
assumes $A = B \ g \in B \rightarrow M \ \bigwedge i. i \in B = \text{simp} \Rightarrow f\ i = g\ i$
shows $\text{fincomp}\ f\ A = \text{fincomp}\ g\ B$
 $\langle \text{proof} \rangle$

Usually, if this rule causes a failed congruence proof error, the reason is that the premise $g \in B \rightarrow M$ cannot be shown. Adding *Pi-def* to the simpset is often useful. For this reason, *fincomp-cong* is not added to the simpset by default.

lemma *fincomp-0* [*simp*]:
 $f \in \{0 :: \text{nat}\} \rightarrow M \Rightarrow \text{fincomp}\ f\ \{..0\} = f\ 0$
 $\langle \text{proof} \rangle$

lemma *fincomp-0'*: $f \in \{..n\} \rightarrow M \Rightarrow (f\ 0) \cdot \text{fincomp}\ f\ \{\text{Suc}\ 0..n\} = \text{fincomp}\ f\ \{..n\}$
 $\langle \text{proof} \rangle$

lemma *fincomp-Suc* [*simp*]:
 $f \in \{.. \text{Suc}\ n\} \rightarrow M \Rightarrow \text{fincomp}\ f\ \{.. \text{Suc}\ n\} = (f\ (\text{Suc}\ n)) \cdot \text{fincomp}\ f\ \{..n\}$
 $\langle \text{proof} \rangle$

lemma *fincomp-Suc2*:
 $f \in \{.. \text{Suc}\ n\} \rightarrow M \Rightarrow \text{fincomp}\ f\ \{.. \text{Suc}\ n\} = (\text{fincomp}\ (\%i. f\ (\text{Suc}\ i))\ \{..n\}) \cdot f\ 0$
 $\langle \text{proof} \rangle$

lemma *fincomp-Suc3*:
assumes $f \in \{..n :: \text{nat}\} \rightarrow M$
shows $\text{fincomp}\ f\ \{.. n\} = (f\ n) \cdot \text{fincomp}\ f\ \{..< n\}$
 $\langle \text{proof} \rangle$

lemma *fincomp-reindex*:
 $f \in (h\ ' A) \rightarrow M \Rightarrow$
 $\text{inj-on}\ h\ A \Rightarrow \text{fincomp}\ f\ (h\ ' A) = \text{fincomp}\ (\lambda x. f\ (h\ x))\ A$
 $\langle \text{proof} \rangle$

lemma *fincomp-const*:
assumes a [*simp*]: $a \in M$
shows $\text{fincomp}\ (\lambda x. a)\ A = \text{rec-nat}\ 1\ (\lambda u. (\cdot)\ a)\ (\text{card}\ A)$
 $\langle \text{proof} \rangle$

lemma *fincomp-singleton*:
assumes $i\text{-in-}A$: $i \in A$ **and** $\text{fin-}A$: $\text{finite}\ A$ **and** $f\text{-Pi}$: $f \in A \rightarrow M$
shows $\text{fincomp}\ (\lambda j. \text{if } i = j \text{ then } f\ j \text{ else } 1)\ A = f\ i$
 $\langle \text{proof} \rangle$

lemma *fincomp-singleton-swap*:
assumes $i\text{-in-}A$: $i \in A$ **and** $\text{fin-}A$: $\text{finite}\ A$ **and** $f\text{-Pi}$: $f \in A \rightarrow M$
shows $\text{fincomp}\ (\lambda j. \text{if } j = i \text{ then } f\ j \text{ else } 1)\ A = f\ i$
 $\langle \text{proof} \rangle$

lemma *fincomp-mono-neutral-cong-left*:

assumes *finite B*
and $A \subseteq B$
and $l: \bigwedge i. i \in B - A \implies h\ i = \mathbf{1}$
and $gh: \bigwedge x. x \in A \implies g\ x = h\ x$
and $h: h \in B \rightarrow M$
shows $\text{fincomp } g\ A = \text{fincomp } h\ B$
 $\langle \text{proof} \rangle$

lemma *fincomp-mono-neutral-cong-right*:

assumes *finite B*
and $A \subseteq B \bigwedge i. i \in B - A \implies g\ i = \mathbf{1} \bigwedge x. x \in A \implies g\ x = h\ x\ g \in B \rightarrow M$
shows $\text{fincomp } g\ B = \text{fincomp } h\ A$
 $\langle \text{proof} \rangle$

lemma *fincomp-mono-neutral-cong*:

assumes [*simp*]: *finite B finite A*
and $*$: $\bigwedge i. i \in B - A \implies h\ i = \mathbf{1} \bigwedge i. i \in A - B \implies g\ i = \mathbf{1}$
and $gh: \bigwedge x. x \in A \cap B \implies g\ x = h\ x$
and $g: g \in A \rightarrow M$
and $h: h \in B \rightarrow M$
shows $\text{fincomp } g\ A = \text{fincomp } h\ B$
 $\langle \text{proof} \rangle$

lemma *fincomp-UN-disjoint*:

assumes
finite I $\bigwedge i. i \in I \implies \text{finite } (A\ i) \text{ pairwise } (\lambda i\ j. \text{disjnt } (A\ i) (A\ j))\ I$
 $\bigwedge i\ x. i \in I \implies x \in A\ i \implies g\ x \in M$
shows $\text{fincomp } g\ (\bigcup (A\ 'I)) = \text{fincomp } (\lambda i. \text{fincomp } g\ (A\ i))\ I$
 $\langle \text{proof} \rangle$

lemma *fincomp-Union-disjoint*:

$\llbracket \text{finite } C; \bigwedge A. A \in C \implies \text{finite } A \wedge (\forall x \in A. f\ x \in M); \text{pairwise disjoint } C \rrbracket \implies$
 $\text{fincomp } f\ (\bigcup C) = \text{fincomp } (\text{fincomp } f)\ C$
 $\langle \text{proof} \rangle$

end

41.2 Results for Abelian Groups

context *Abelian-Group* **begin**

lemma *fincomp-inverse*:

$f \in A \rightarrow G \implies \text{fincomp } (\lambda x. \text{inverse } (f\ x))\ A = \text{inverse } (\text{fincomp } f\ A)$
 $\langle \text{proof} \rangle$

Jeremy Avigad. This should be generalized to arbitrary groups, not just Abelian ones, using Lagrange's theorem.

lemma *power-order-eq-one*:

```

assumes fin [simp]: finite G
and a [simp]: a ∈ G
shows rec-nat 1 ( $\lambda u. (\cdot) a$ ) (card G) = 1
⟨proof⟩

end

```

41.3 Transfer between two commutative monoids

commutative-monoid.fincomp depends on the *carrier* as well as on the operation, through *commutative-monoid.M-ify*, so two structures whose operations agree do not automatically have the same finite composites, and *commutative-monoid.fincomp-cong'* — which stays within one monoid — does not apply. What is true is the following: if the smaller carrier is contained in the larger, the operations agree there, the unit is shared, and the function composed takes its values in the smaller carrier, then the composites agree. This is what allows a sum computed inside a substructure to be read in the ambient structure, and in particular what makes evaluation of a polynomial independent of which of two nested structures it is performed in.

The other direction of the same gap: a finite composite may be pushed *along a homomorphism*. HOL-Algebra gets the corresponding fact for polynomial evaluation free from *ring-hom-ring.eval-hom'*; the carrier-set layer has no homomorphism apparatus at this level, so it is proved here directly, by the same finite induction as the transfer lemma below.

lemma *fincomp-hom*:

```

assumes M: commutative-monoid M f e and N: commutative-monoid N g d
and hM:  $\bigwedge x. x \in M \implies h\ x \in N$ 
and he: h e = d
and hf:  $\bigwedge x\ y. \llbracket x \in M; y \in M \rrbracket \implies h\ (f\ x\ y) = g\ (h\ x)\ (h\ y)$ 
and k: k ∈ A → M
shows h (commutative-monoid.fincomp M f e k A)
      = commutative-monoid.fincomp N g d ( $\lambda i. h\ (k\ i)$ ) A
⟨proof⟩

```

lemma *fincomp-transfer*:

```

assumes M: commutative-monoid M f e and N: commutative-monoid N g e
and sub: M ⊆ N
and agree:  $\bigwedge x\ y. \llbracket x \in M; y \in M \rrbracket \implies f\ x\ y = g\ x\ y$ 
and h: h ∈ A → M
shows commutative-monoid.fincomp M f e h A = commutative-monoid.fincomp N
      g e h A
⟨proof⟩

```

end

42 The Polynomial Ring $R[X]$ (Carrier-Set Construction)

```
theory Poly-Ring
  imports Ring-Theory Finite-Composite
begin
```

Note on notation. *Ring-Theory* frees $+/-$ for the ring operations, while *FiniteProduct* (via *Main*) keeps HOL's arithmetic notation; the theory merge admits *both*. This is exactly what the convolution needs — ring operations on coefficients (type $'a$) and ordinary arithmetic on the *nat* indices $k - i$.

We construct the polynomial ring over a carrier-set ring R using the *finite-support function* representation: a polynomial is a function $\text{nat} \Rightarrow 'a$ that is eventually zero, with all coefficients drawn from the carrier R . This mirrors HOL's own $'a$ *poly* type (which is *Abs-poly* over a finite-support function), so the bridge to the type-class world is near definitional, and there is no leading-zero normalisation invariant to maintain.

Coefficient i of a polynomial p is simply $p\ i$. Addition is pointwise; multiplication is the convolution $(p \otimes q)\ k = (\sum_{i \leq k}. p\ i \cdot q\ (k - i))$, computed with the additive *fincomp* operator of R .

```
context Ring
begin
```

42.1 Carrier, coefficients, and support

The set of polynomials over R : finite support, coefficients in R .

```
definition poly-carrier :: (nat  $\Rightarrow$  'a) set
  where poly-carrier  $\equiv$  {p. finite {i. p i  $\neq$  0}  $\wedge$  ( $\forall$  i. p i  $\in$  R)}
```

```
lemma poly-carrierI:
  [| finite {i. p i  $\neq$  0};  $\wedge$  i. p i  $\in$  R |]  $\implies$  p  $\in$  poly-carrier
  <proof>
```

```
lemma poly-carrier-finite: p  $\in$  poly-carrier  $\implies$  finite {i. p i  $\neq$  0}
  <proof>
```

```
lemma poly-carrier-coeff-closed: p  $\in$  poly-carrier  $\implies$  p i  $\in$  R
  <proof>
```

42.2 Zero and one

```
definition poly-zero :: nat  $\Rightarrow$  'a ( $\langle 0_P \rangle$ )
  where poly-zero  $\equiv$  ( $\lambda$  i. 0)
```

```
definition poly-one :: nat  $\Rightarrow$  'a ( $\langle 1_P \rangle$ )
  where poly-one  $\equiv$  ( $\lambda$  i. if i = 0 then 1 else 0)
```

42.3 Addition

definition $\text{poly-add} :: (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a)$ (**infixl** $\langle \oplus_P \rangle$ 65)
where $\text{poly-add } p \ q \equiv (\lambda i. \ p \ i + q \ i)$

definition $\text{poly-neg} :: (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a)$ ($\langle \ominus_P \rangle$ [66] 65)
where $\text{poly-neg } p \equiv (\lambda i. \ - \ p \ i)$

42.4 Multiplication (convolution)

definition $\text{poly-mult} :: (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a)$ (**infixl** $\langle \otimes_P \rangle$ 70)
where $\text{poly-mult } p \ q \equiv (\lambda k. \ \text{additive.fincomp } (\lambda i. \ p \ i \cdot q \ (k - i)) \ \{..k\})$

A polynomial is a *unit* when it has a two-sided multiplicative inverse in the polynomial ring. (Stated directly on the carrier-set operations, since the ambient *multiplicative* monoid acts on the coefficient type, not on polynomials.)

definition $\text{poly-unit} :: (\text{nat} \Rightarrow 'a) \Rightarrow \text{bool}$
where $\text{poly-unit } p \equiv p \in \text{poly-carrier}$
 $\wedge (\exists q. \ q \in \text{poly-carrier} \wedge p \otimes_P q = \mathbf{1}_P \wedge q \otimes_P p = \mathbf{1}_P)$

lemma poly-unitI :

$\llbracket p \in \text{poly-carrier}; q \in \text{poly-carrier}; p \otimes_P q = \mathbf{1}_P; q \otimes_P p = \mathbf{1}_P \rrbracket \Longrightarrow \text{poly-unit } p$
 $\langle \text{proof} \rangle$

lemma poly-unit-closed : $\text{poly-unit } p \Longrightarrow p \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

42.5 Closure under the operations

lemma poly-zero-closed : $\mathbf{0}_P \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

lemma poly-one-closed : $\mathbf{1}_P \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

lemma poly-add-closed :

assumes $p \in \text{poly-carrier} \ q \in \text{poly-carrier}$

shows $p \oplus_P q \in \text{poly-carrier}$

$\langle \text{proof} \rangle$

Each coefficient of a product is a finite sum of products of carrier elements.

lemma $\text{poly-mult-coeff-closed}$:

assumes $p \in \text{poly-carrier} \ q \in \text{poly-carrier}$

shows $(p \otimes_P q) \ k \in R$

$\langle \text{proof} \rangle$

A product coefficient vanishes beyond the sum of the two top degrees, so the product has finite support.

lemma $\text{poly-mult-support-bound}$:

assumes $p: p \in \text{poly-carrier}$ **and** $q: q \in \text{poly-carrier}$
and $k: k > (\text{if } \{i. p \ i \neq \mathbf{0}\} = \{\} \text{ then } 0 \text{ else } \text{Max } \{i. p \ i \neq \mathbf{0}\})$
 $+ (\text{if } \{i. q \ i \neq \mathbf{0}\} = \{\} \text{ then } 0 \text{ else } \text{Max } \{i. q \ i \neq \mathbf{0}\})$
shows $(p \otimes_P q) \ k = \mathbf{0}$
 $\langle \text{proof} \rangle$

lemma *poly-mult-closed*:
assumes $p \in \text{poly-carrier}$ $q \in \text{poly-carrier}$
shows $p \otimes_P q \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

42.6 Distribution of multiplication over finite additive sums

A fixed factor distributes over an additive finite sum (left and right versions).

lemma *fincomp-mult-distrib-left*:
assumes $a \in R$ **and** $g \in A \rightarrow R$
shows $a \cdot \text{additive.fincomp } g \ A = \text{additive.fincomp } (\lambda i. a \cdot g \ i) \ A$
 $\langle \text{proof} \rangle$

lemma *fincomp-mult-distrib-right*:
assumes $a \in R$ **and** $g \in A \rightarrow R$
shows $\text{additive.fincomp } g \ A \cdot a = \text{additive.fincomp } (\lambda i. g \ i \cdot a) \ A$
 $\langle \text{proof} \rangle$

42.7 Pointwise additive structure

lemma *poly-add-assoc*:
 $\llbracket p \in \text{poly-carrier}; q \in \text{poly-carrier}; r \in \text{poly-carrier} \rrbracket$
 $\implies (p \oplus_P q) \oplus_P r = p \oplus_P (q \oplus_P r)$
 $\langle \text{proof} \rangle$

lemma *poly-add-comm*:
 $\llbracket p \in \text{poly-carrier}; q \in \text{poly-carrier} \rrbracket \implies p \oplus_P q = q \oplus_P p$
 $\langle \text{proof} \rangle$

lemma *poly-add-zero*: $p \in \text{poly-carrier} \implies \mathbf{0}_P \oplus_P p = p$
 $\langle \text{proof} \rangle$

lemma *poly-neg-closed*: $p \in \text{poly-carrier} \implies \ominus_P p \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

lemma *poly-add-neg*: $p \in \text{poly-carrier} \implies (\ominus_P p) \oplus_P p = \mathbf{0}_P$
 $\langle \text{proof} \rangle$

lemma *poly-add-neg-right*: $p \in \text{poly-carrier} \implies p \oplus_P (\ominus_P p) = \mathbf{0}_P$
 $\langle \text{proof} \rangle$

Additive cancellation: $x \oplus (a \oplus \ominus x) = a$ for carrier polynomials.

lemma *poly-add-minus-cancel*:

assumes $x \in \text{poly-carrier}$ $a \in \text{poly-carrier}$
shows $x \oplus_P (a \oplus_P (\ominus_P x)) = a$
 $\langle \text{proof} \rangle$

Left cancellation for polynomial addition.

lemma *poly-add-left-cancel*:
assumes $x: x \in \text{poly-carrier}$ **and** $a: a \in \text{poly-carrier}$ **and** $b: b \in \text{poly-carrier}$
and $eq: x \oplus_P a = x \oplus_P b$
shows $a = b$
 $\langle \text{proof} \rangle$

The negation is the unique additive inverse.

lemma *poly-minus-unique*:
assumes $x: x \in \text{poly-carrier}$ **and** $y: y \in \text{poly-carrier}$ **and** $z: x \oplus_P y = \mathbf{0}_P$
shows $y = \ominus_P x$
 $\langle \text{proof} \rangle$

Double negation.

lemma *poly-neg-neg*:
assumes $x: x \in \text{poly-carrier}$ **shows** $\ominus_P (\ominus_P x) = x$
 $\langle \text{proof} \rangle$

Reassociation: $(u \oplus a) \oplus c = (u \oplus c) \oplus a$ in the abelian group.

lemma *poly-add-swap-right*:
assumes $u: u \in \text{poly-carrier}$ **and** $a: a \in \text{poly-carrier}$ **and** $c: c \in \text{poly-carrier}$
shows $(u \oplus_P a) \oplus_P c = (u \oplus_P c) \oplus_P a$
 $\langle \text{proof} \rangle$

Subtracting one equation from another: if $u \oplus a = v \oplus b$ then $u \oplus \ominus v = b \oplus \ominus a$.

lemma *poly-sub-shift*:
assumes $u: u \in \text{poly-carrier}$ **and** $v: v \in \text{poly-carrier}$
and $a: a \in \text{poly-carrier}$ **and** $b: b \in \text{poly-carrier}$
and $eq: u \oplus_P a = v \oplus_P b$
shows $u \oplus_P (\ominus_P v) = b \oplus_P (\ominus_P a)$
 $\langle \text{proof} \rangle$

If $x \oplus \ominus y = \mathbf{0}$ then $x = y$.

lemma *poly-diff-zero-eq*:
assumes $x \in \text{poly-carrier}$ **and** $y \in \text{poly-carrier}$ **and** $x \oplus_P (\ominus_P y) = \mathbf{0}_P$
shows $x = y$
 $\langle \text{proof} \rangle$

42.8 Multiplicative unit

lemma *poly-mult-one-left*:
assumes $p: p \in \text{poly-carrier}$
shows $\mathbf{1}_P \otimes_P p = p$

<proof>

lemma *poly-mult-one-right*:
assumes $p: p \in \text{poly-carrier}$
shows $p \otimes_P \mathbf{1}_P = p$
<proof>

42.9 Distributivity

lemma *poly-mult-add-distrib-left*:
assumes $p: p \in \text{poly-carrier}$ **and** $q: q \in \text{poly-carrier}$ **and** $r: r \in \text{poly-carrier}$
shows $p \otimes_P (q \oplus_P r) = (p \otimes_P q) \oplus_P (p \otimes_P r)$
<proof>

lemma *poly-mult-add-distrib-right*:
assumes $p: p \in \text{poly-carrier}$ **and** $q: q \in \text{poly-carrier}$ **and** $r: r \in \text{poly-carrier}$
shows $(q \oplus_P r) \otimes_P p = (q \otimes_P p) \oplus_P (r \otimes_P p)$
<proof>

42.10 A Fubini law for the additive finite sum

A nested additive sum equals the sum over the dependent product (Sigma) of the index sets: $\sum_{i \in I}. \sum_{j \in A \ i}. g \ i \ j = \sum_{(i,j) \in \text{Sigma } I \ A}. g \ i \ j.$

lemma *fincomp-Sigma*:
assumes $I: \text{finite } I$ **and** $A: \bigwedge i. i \in I \implies \text{finite } (A \ i)$
and $g: \bigwedge i \ j. i \in I \implies j \in A \ i \implies g \ i \ j \in R$
shows $\text{additive.fincomp } (\lambda i. \text{additive.fincomp } (g \ i) (A \ i)) \ I$
 $= \text{additive.fincomp } (\lambda p. g \ (\text{fst } p) \ (\text{snd } p)) \ (\text{Sigma } I \ A)$
<proof>

42.11 Associativity of convolution

lemma *poly-mult-assoc*:
assumes $p: p \in \text{poly-carrier}$ **and** $q: q \in \text{poly-carrier}$ **and** $r: r \in \text{poly-carrier}$
shows $(p \otimes_P q) \otimes_P r = p \otimes_P (q \otimes_P r)$
<proof>

42.12 Coefficients, degree, monomials, constants, and the variable

The coefficient map is just function application (kept as a named operation to mirror the HOL-Algebra polynomial interface).

definition *coeff* :: $(\text{nat} \Rightarrow 'a) \Rightarrow \text{nat} \Rightarrow 'a$
where $\text{coeff } p = p$

lemma *coeff-eq [simp]*: $\text{coeff } p \ i = p \ i$
<proof>

The degree is the greatest index with a nonzero coefficient (and 0 for the zero polynomial, as is conventional here).

definition *degree* :: (nat $\Rightarrow$ 'a) $\Rightarrow$ nat
where *degree* *p* $\equiv$ (if {*i*. *p* *i* $\neq$ 0} = {} then 0 else Max {*i*. *p* *i* $\neq$ 0})

lemma *degree-zero* [simp]: *degree* 0_P = 0
 <proof>

lemma *coeff-gt-degree*:
assumes *p* $\in$ *poly-carrier* **and** *degree* *p* < *i*
shows *p* *i* = 0
 <proof>

A degree bound: *degree* *p* $\leq$ *n* exactly when all coefficients above *n* vanish (for a polynomial, whose support is finite).

lemma *degree-le-iff*:
assumes *p* $\in$ *poly-carrier*
shows *degree* *p* $\leq$ *n* $\longleftrightarrow$ ($\forall i > n$. *p* *i* = 0)
 <proof>

lemma *degree-leI*:
 $\llbracket p \in \text{poly-carrier}; \bigwedge i. i > n \implies p\ i = 0 \rrbracket \implies \text{degree } p \leq n$
 <proof>

Negation preserves the support, hence the degree (for carrier polynomials).

lemma *degree-neg*:
assumes *p*: *p* $\in$ *poly-carrier* **shows** *degree* ($\ominus_P$ *p*) = *degree* *p*
 <proof>

42.13 Polynomials of bounded degree

The polynomials supported on $\{..<n\}$ — those of degree below *n*, together with the zero polynomial. These are exactly the carrier polynomials whose coefficients vanish from index *n* on. When the coefficient ring is finite, there are $|R|^n$ of them; they serve as the canonical representatives of $F[X]/(b)$ for *b* of degree *n*.

definition *low-poly* :: nat $\Rightarrow$ (nat $\Rightarrow$ 'a) set
where *low-poly* *n* $\equiv$ {*p*. *p* $\in$ *poly-carrier* $\wedge$ ($\forall i \geq n$. *p* *i* = 0)}

lemma *low-polyI*:
 $\llbracket p \in \text{poly-carrier}; \bigwedge i. i \geq n \implies p\ i = 0 \rrbracket \implies p \in \text{low-poly } n$
 <proof>

lemma *low-poly-closed*: *p* $\in$ *low-poly* *n* $\implies$ *p* $\in$ *poly-carrier*
 <proof>

lemma *low-poly-coeff*: $\llbracket p \in \text{low-poly } n; i \geq n \rrbracket \implies p\ i = 0$
 <proof>

The low-degree polynomials form a subgroup of the additive group, and this is immediate *coefficientwise*: the operations are pointwise, so no reasoning about degrees is needed.

lemma *low-poly-zero* [simp]: $\mathbf{0}_P \in \text{low-poly } n$
 $\langle \text{proof} \rangle$

lemma *low-poly-add-closed*:
assumes $p: p \in \text{low-poly } n$ **and** $q: q \in \text{low-poly } n$ **shows** $p \oplus_P q \in \text{low-poly } n$
 $\langle \text{proof} \rangle$

lemma *low-poly-neg-closed*:
assumes $p: p \in \text{low-poly } n$ **shows** $\ominus_P p \in \text{low-poly } n$
 $\langle \text{proof} \rangle$

lemma *low-poly-one*:
assumes $n: 0 < n$ **shows** $\mathbf{1}_P \in \text{low-poly } n$
 $\langle \text{proof} \rangle$

Membership of *low-poly* in terms of degree: a carrier polynomial lies in *low-poly* n iff it is zero or has degree below n .

lemma *low-poly-iff-degree*:
assumes $p: p \in \text{poly-carrier}$
shows $p \in \text{low-poly } n \longleftrightarrow (p = \mathbf{0}_P \vee \text{degree } p < n)$
 $\langle \text{proof} \rangle$

Restricting a low-degree polynomial to $\{..<n\}$ is a bijection onto the coefficient vectors $\{..<n\} \rightarrow_E R$; the inverse extends a vector by zero.

lemma *bij-low-poly-PiE*:
bij-betw $(\lambda p. \text{restrict } p \{..<n\}) (\text{low-poly } n) (\{..<n\} \rightarrow_E R)$
 $\langle \text{proof} \rangle$

Hence, over a finite coefficient ring, there are exactly $|R|^n$ polynomials of degree below n .

theorem *card-low-poly*:
assumes *finite* R **shows** $\text{card } (\text{low-poly } n) = \text{card } R \wedge^n$
 $\langle \text{proof} \rangle$

The constant polynomial with value a .

definition *poly-const* :: $'a \Rightarrow (\text{nat} \Rightarrow 'a)$
where $\text{poly-const } a = (\lambda i. \text{if } i = 0 \text{ then } a \text{ else } \mathbf{0})$

lemma *poly-const-closed*: $a \in R \implies \text{poly-const } a \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

lemma *poly-const-zero* [simp]: $\text{poly-const } \mathbf{0} = \mathbf{0}_P$
 $\langle \text{proof} \rangle$

lemma *poly-const-one* [simp]: $\text{poly-const } \mathbf{1} = \mathbf{1}_P$

$\langle proof \rangle$

The monomial $a X^n$: the coefficient a at degree n .

definition $monom :: 'a \Rightarrow nat \Rightarrow (nat \Rightarrow 'a)$
where $monom\ a\ n = (\lambda i. \text{if } i = n \text{ then } a \text{ else } \mathbf{0})$

lemma $monom-closed$: $a \in R \implies monom\ a\ n \in poly-carrier$
 $\langle proof \rangle$

lemma $monom-0-eq-const$: $monom\ a\ 0 = poly-const\ a$
 $\langle proof \rangle$

The variable X is the monomial $\mathbf{1}\ X^1$.

definition $var :: nat \Rightarrow 'a\ (\langle X_P \rangle)$
where $var = monom\ \mathbf{1}\ 1$

lemma $var-closed$: $X_P \in poly-carrier$
 $\langle proof \rangle$

lemma $coeff-var$: $X_P\ i = (\text{if } i = 1 \text{ then } \mathbf{1} \text{ else } \mathbf{0})$
 $\langle proof \rangle$

Multiplying by a monomial $monom\ c\ k$ shifts coefficients up by k and scales by c : $(monom\ c\ k \otimes b)\ j = (\text{if } k \leq j \text{ then } c \cdot b\ (j - k) \text{ else } \mathbf{0})$.

lemma $coeff-monom-mult$:
assumes $c: c \in R$ **and** $b: b \in poly-carrier$
shows $(monom\ c\ k \otimes_P b)\ j = (\text{if } k \leq j \text{ then } c \cdot b\ (j - k) \text{ else } \mathbf{0})$
 $\langle proof \rangle$

Degrees of the basic polynomials

Multiplying a monomial by the variable raises its exponent, and multiplying by a constant scales its coefficient.

lemma $var-mult-monom$:
assumes $a: a \in R$ **shows** $X_P \otimes_P monom\ a\ k = monom\ a\ (Suc\ k)$
 $\langle proof \rangle$

A monomial is zero exactly when its coefficient is zero.

lemma $monom-eq-zero-iff$ $[simp]$: $monom\ a\ n = \mathbf{0}_P \longleftrightarrow a = \mathbf{0}$
 $\langle proof \rangle$

lemma $degree-monom$:
assumes $a \neq \mathbf{0}$
shows $degree\ (monom\ a\ n) = n$
 $\langle proof \rangle$

lemma $degree-monom-le$: $degree\ (monom\ a\ n) \leq n$
 $\langle proof \rangle$

lemma *degree-poly-const*:
assumes $a \neq \mathbf{0}$
shows $\text{degree } (\text{poly-const } a) = 0$
 $\langle \text{proof} \rangle$

lemma *degree-const-le*: $\text{degree } (\text{poly-const } a) \leq 0$
 $\langle \text{proof} \rangle$

lemma *degree-one* [simp]: $\text{degree } \mathbf{1}_P = 0$
 $\langle \text{proof} \rangle$

In a nontrivial coefficient ring, the variable has degree one. The explicit hypothesis keeps this lemma available in the more general *Ring* context above.

lemma *degree-var*:
assumes $\mathbf{1} \neq \mathbf{0}$
shows $\text{degree } X_P = 1$
 $\langle \text{proof} \rangle$

The product of two constants is the constant of the product.

lemma *poly-const-add*:
assumes $a: a \in R$ **and** $b: b \in R$
shows $\text{poly-const } a \oplus_P \text{poly-const } b = \text{poly-const } (a + b)$
 $\langle \text{proof} \rangle$

lemma *poly-const-mult*:
assumes $a: a \in R$ **and** $b: b \in R$
shows $\text{poly-const } a \otimes_P \text{poly-const } b = \text{poly-const } (a \cdot b)$
 $\langle \text{proof} \rangle$

A polynomial of degree 0 is the constant given by its zeroth coefficient.

lemma *degree-zero-imp-const*:
assumes $p \in \text{poly-carrier}$ $\text{degree } p = 0$
shows $p = \text{poly-const } (p \ 0)$
 $\langle \text{proof} \rangle$

lemma *degree-add-le*:
assumes $p \in \text{poly-carrier}$ $q \in \text{poly-carrier}$
shows $\text{degree } (p \oplus_P q) \leq \max (\text{degree } p) (\text{degree } q)$
 $\langle \text{proof} \rangle$

Multiplication by the zero polynomial.

lemma
assumes $p \in \text{poly-carrier}$
shows *poly-mult-zero-right*: $p \otimes_P \mathbf{0}_P = \mathbf{0}_P$
and *poly-mult-zero-left*: $\mathbf{0}_P \otimes_P p = \mathbf{0}_P$
 $\langle \text{proof} \rangle$

The leading coefficient.

definition *lead-coeff* :: (nat $\Rightarrow$ 'a) $\Rightarrow$ 'a
where *lead-coeff* p = p (degree p)

The coefficient at the degree is nonzero for a nonzero polynomial.

lemma *coeff-degree-nonzero*:
assumes p $\in$ poly-carrier **and** p $\neq$ 0_P
shows p (degree p) $\neq$ 0
 <proof>

lemma *lead-coeff-nonzero*:
 $\llbracket p \in \text{poly-carrier}; p \neq 0_P \rrbracket \implies \text{lead-coeff } p \neq 0$
 <proof>

The coefficient of a product at the sum of the two degrees is the product of the leading coefficients: every other term of the convolution vanishes for degree reasons.

lemma *coeff-mult-degree-add*:
assumes p: p $\in$ poly-carrier **and** q: q $\in$ poly-carrier
shows (p $\otimes_P$ q) (degree p + degree q) = p (degree p) $\cdot$ q (degree q)
 <proof>

The degree of a product is at most the sum of the degrees.

lemma *degree-mult-le*:
assumes p: p $\in$ poly-carrier **and** q: q $\in$ poly-carrier
shows degree (p $\otimes_P$ q) $\leq$ degree p + degree q
 <proof>

42.14 The ring structure

theorem *poly-ring*: Ring poly-carrier ($\oplus_P$) ($\otimes_P$) 0_P 1_P
 <proof>

42.15 Evaluation at a point

Powers in the (multiplicative monoid of the) ring. We define them locally rather than importing *Group-Action*'s *Monoid.power*, to keep the polynomial theory's dependencies minimal.

definition *rpow* :: 'a $\Rightarrow$ nat $\Rightarrow$ 'a
where *rpow* a n = ((($\cdot$) a) $\wedge^{\wedge}$ n) 1

lemma *rpow-0 [simp]*: *rpow* a 0 = 1
 <proof>

lemma *rpow-Suc [simp]*: *rpow* a (Suc n) = a $\cdot$ *rpow* a n
 <proof>

lemma *rpow-closed [simp]*: a $\in$ R $\implies$ *rpow* a n $\in$ R
 <proof>

Evaluation of a polynomial at a point a : the finite sum $\sum_{i \leq \text{degree } p} p \ i \cdot a^i$.

definition $\text{eval} :: 'a \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow 'a$
where $\text{eval } a \ p = \text{additive.fincomp } (\lambda i. p \ i \cdot \text{rpow } a \ i) \ \{\dots \text{degree } p\}$

Each summand lies in R , and the sum is closed.

lemma $\text{eval-summand-closed}$: $\llbracket p \in \text{poly-carrier}; a \in R \rrbracket \Longrightarrow p \ i \cdot \text{rpow } a \ i \in R$
 $\langle \text{proof} \rangle$

lemma eval-closed $[\text{simp}]$:
assumes $p: p \in \text{poly-carrier}$ **and** $a: a \in R$ **shows** $\text{eval } a \ p \in R$
 $\langle \text{proof} \rangle$

The summation range may be enlarged past the degree: the extra coefficients vanish.

lemma eval-fincomp-le :
assumes $p: p \in \text{poly-carrier}$ **and** $a: a \in R$ **and** $n: \text{degree } p \leq n$
shows $\text{eval } a \ p = \text{additive.fincomp } (\lambda i. p \ i \cdot \text{rpow } a \ i) \ \{\dots n\}$
 $\langle \text{proof} \rangle$

Evaluation of the basic polynomials.

lemma eval-zero $[\text{simp}]$: $a \in R \Longrightarrow \text{eval } a \ \mathbf{0}_P = \mathbf{0}$
 $\langle \text{proof} \rangle$

lemma eval-const $[\text{simp}]$:
assumes $a: a \in R$ **and** $c: c \in R$ **shows** $\text{eval } a \ (\text{poly-const } c) = c$
 $\langle \text{proof} \rangle$

lemma eval-one $[\text{simp}]$: $a \in R \Longrightarrow \text{eval } a \ \mathbf{1}_P = \mathbf{1}$
 $\langle \text{proof} \rangle$

Evaluation is additive.

lemma eval-add :
assumes $p: p \in \text{poly-carrier}$ **and** $q: q \in \text{poly-carrier}$ **and** $a: a \in R$
shows $\text{eval } a \ (p \oplus_P q) = \text{eval } a \ p + \text{eval } a \ q$
 $\langle \text{proof} \rangle$

Evaluation of a monomial $\text{monom } c \ k$ is $c \cdot a^k$.

lemma eval-monom :
assumes $c: c \in R$ **and** $a: a \in R$
shows $\text{eval } a \ (\text{monom } c \ k) = c \cdot \text{rpow } a \ k$
 $\langle \text{proof} \rangle$

Evaluation of the variable X_P returns the point itself.

lemma poly-rpow-var : $\text{Ring.rpow } (\otimes_P) \ \mathbf{1}_P \ X_P \ k = \text{monom } \mathbf{1} \ k$
 $\langle \text{proof} \rangle$

lemma $\text{poly-const-mult-monom}$:

assumes $c: c \in R$ **shows** $\text{poly-const } c \otimes_P \text{monom } \mathbf{1} \ k = \text{monom } c \ k$
 $\langle \text{proof} \rangle$

Expanding a polynomial into its monomials. Sums in the polynomial ring are pointwise, so a finite composite may be read coefficientwise, and then only the monomial whose exponent matches the coefficient index survives. This is the univariate counterpart of the closed form *ieval-ivar* in *Indexed-Poly*, and it is what identifies evaluation of a polynomial's constant lift at the variable with the polynomial itself.

lemma *poly-add-cmonoid: commutative-monoid poly-carrier* $(\oplus_P) \ \mathbf{0}_P$
 $\langle \text{proof} \rangle$

lemma *poly-fincomp-apply:*
assumes $f: \bigwedge k. f \ k \in \text{poly-carrier}$
shows $\text{commutative-monoid.fincomp poly-carrier } (\oplus_P) \ \mathbf{0}_P \ f \ A \ j$
 $= \text{additive.fincomp } (\lambda k. f \ k \ j) \ A$
 $\langle \text{proof} \rangle$

lemma *poly-sum-monom:*
assumes $q: q \in \text{poly-carrier}$ **and** $n: \text{degree } q \leq n$
shows $\text{commutative-monoid.fincomp poly-carrier } (\oplus_P) \ \mathbf{0}_P \ (\lambda k. \text{monom } (q \ k) \ k)$
 $\{..n\} = q$
 $\langle \text{proof} \rangle$

A polynomial is the value of its own constant lift at the variable. Evaluating in the polynomial ring itself, with each coefficient embedded as a constant and the point taken to be X_P , returns the polynomial. Composing this with the reduction homomorphism is what identifies the class of q with the value of q at the adjoined root.

lemma *eval-const-lift-at-var:*
assumes $q: q \in \text{poly-carrier}$
shows $\text{Ring.eval poly-carrier } (\oplus_P) \ (\otimes_P) \ \mathbf{0}_P \ \mathbf{1}_P \ X_P \ (\lambda k. \text{poly-const } (q \ k)) = q$
 $\langle \text{proof} \rangle$

lemma *eval-var [simp]:* $a \in R \implies \text{eval } a \ X_P = a$
 $\langle \text{proof} \rangle$

Powers add exponents.

lemma *rpow-add:* $a \in R \implies \text{rpow } a \ (i + j) = \text{rpow } a \ i \cdot \text{rpow } a \ j$
 $\langle \text{proof} \rangle$

end

42.16 Evaluation along a ring homomorphism

Evaluating in the target, at the image of the point and with the images of the coefficients, gives the image of the evaluation. Powers transport by induction on the exponent and the sum by *fincomp-hom*; the summation range is the only

thing that does not come for free, so the equality of the two degrees is taken as a hypothesis..

lemma *eval-hom*:

```

fixes A :: 'b set and add mult :: 'b  $\Rightarrow$  'b  $\Rightarrow$  'b and z u :: 'b
and B :: 'c set and badd bmult :: 'c  $\Rightarrow$  'c  $\Rightarrow$  'c and bz bu :: 'c
and h :: 'b  $\Rightarrow$  'c
assumes RA: Ring A add mult z u and RB: Ring B badd bmult bz bu
and hB:  $\bigwedge y. y \in A \implies h y \in B$ 
and hz:  $h z = bz$  and hu:  $h u = bu$ 
and hadd:  $\bigwedge y y'. \llbracket y \in A; y' \in A \rrbracket \implies h (add y y') = badd (h y) (h y')$ 
and hmult:  $\bigwedge y y'. \llbracket y \in A; y' \in A \rrbracket \implies h (mult y y') = bmult (h y) (h y')$ 
and x:  $x \in A$  and p:  $\bigwedge k. p k \in A$ 
and deg:  $Ring.degree bz (\lambda k. h (p k)) = Ring.degree z p$ 
shows  $Ring.eval B badd bmult bz bu (h x) (\lambda k. h (p k)) = h (Ring.eval A add mult z u x p)$ 
<proof>

```

When the map is injective on the source and the target is exactly the image, the degree hypothesis is automatic: an injection sends the zero to the zero and nothing else there, so the two coefficient supports are the same set.

corollary *eval-transport*:

```

fixes A :: 'b set and add mult :: 'b  $\Rightarrow$  'b  $\Rightarrow$  'b and z u :: 'b
and h :: 'b  $\Rightarrow$  'c and hadd hmult :: 'c  $\Rightarrow$  'c  $\Rightarrow$  'c
assumes RA: Ring A add mult z u
and RK: Ring (h ' A) hadd hmult (h z) (h u)
and ha:  $\bigwedge y y'. \llbracket y \in A; y' \in A \rrbracket \implies h (add y y') = hadd (h y) (h y')$ 
and hm:  $\bigwedge y y'. \llbracket y \in A; y' \in A \rrbracket \implies h (mult y y') = hmult (h y) (h y')$ 
and inj: inj-on h A
and x:  $x \in A$  and p:  $\bigwedge k. p k \in A$ 
shows  $Ring.eval (h ' A) hadd hmult (h z) (h u) (h x) (\lambda k. h (p k)) = h (Ring.eval A add mult z u x p)$ 
<proof>

```

43 Polynomials over a Commutative Ring (continued): evaluation is multiplicative

context *commutative-ring*
begin

Evaluation is multiplicative. This needs commutativity (the point and the coefficients must commute). The proof mirrors *poly-mult-assoc*: expand both sides to sums over a triangle of index pairs and reindex by $(k, i) \mapsto (i, k - i)$.

lemma *eval-mult*:

```

assumes p:  $p \in poly-carrier$  and q:  $q \in poly-carrier$  and a:  $a \in R$ 
shows  $eval a (p \otimes_P q) = eval a p \cdot eval a q$ 
<proof>

```

Evaluation respects negation, hence is a ring homomorphism $F[X] \rightarrow F$.

lemma *eval-neg*:
assumes $p \in \text{poly-carrier}$ **and** $a \in R$
shows $\text{eval } a (\ominus_P p) = - \text{eval } a p$
 $\langle \text{proof} \rangle$

The linear polynomial $X - a$ (a root factor).

definition *root-factor* :: $'a \Rightarrow (\text{nat} \Rightarrow 'a)$
where $\text{root-factor } a \equiv X_P \oplus_P (\ominus_P \text{poly-const } a)$

lemma *root-factor-closed*: $a \in R \implies \text{root-factor } a \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

root-factor a evaluates to 0 at a .

lemma *eval-root-factor [simp]*:
assumes $a: a \in R$ **shows** $\text{eval } a (\text{root-factor } a) = 0$
 $\langle \text{proof} \rangle$

end

44 Polynomials over a Commutative Ring

context *commutative-ring*
begin

When the coefficient ring is commutative, so is the polynomial ring: the convolution $\sum_{i \leq k} p \ i \cdot q \ (k - i)$ is symmetric under the reindexing $i \mapsto k - i$.

lemma *poly-mult-comm*:
assumes $p: p \in \text{poly-carrier}$ **and** $q: q \in \text{poly-carrier}$
shows $p \otimes_P q = q \otimes_P p$
 $\langle \text{proof} \rangle$

Hence the polynomial ring over a commutative ring is itself commutative.

theorem *poly-commutative-ring*:
 $\text{commutative-ring } \text{poly-carrier } (\oplus_P) (\otimes_P) 0_P 1_P$
 $\langle \text{proof} \rangle$

end

45 Polynomials over an Integral Domain

context *integral-domain*
begin

The leading coefficients are nonzero, and a domain has no zero divisors, so the product's coefficient at $\text{degree } p + \text{degree } q$ is nonzero.

lemma *degree-mult*:

assumes $p: p \in \text{poly-carrier}$ **and** $q: q \in \text{poly-carrier}$
and $pnz: p \neq \mathbf{0}_P$ **and** $qnz: q \neq \mathbf{0}_P$
shows $\text{degree } (p \otimes_P q) = \text{degree } p + \text{degree } q$
 $\langle \text{proof} \rangle$

Multiplication by a nonzero monomial shifts the degree by its exponent.

lemma *degree-monom-mult:*
assumes $c: c \in R$ **and** $b: b \in \text{poly-carrier}$
and $cnz: c \neq \mathbf{0}$ **and** $bnz: b \neq \mathbf{0}_P$
shows $\text{degree } (\text{monom } c \ n \otimes_P b) = n + \text{degree } b$
 $\langle \text{proof} \rangle$

Hence the polynomial ring over an integral domain has no zero divisors.

lemma *poly-no-zero-divisors:*
assumes $p \in \text{poly-carrier}$ **and** $q \in \text{poly-carrier}$ **and** $p \otimes_P q = \mathbf{0}_P$
shows $p = \mathbf{0}_P \vee q = \mathbf{0}_P$
 $\langle \text{proof} \rangle$

Hence $R[X]$ is itself an integral domain. .

theorem *poly-integral-domain:*
integral-domain poly-carrier $(\oplus_P) (\otimes_P) \mathbf{0}_P \mathbf{1}_P$
 $\langle \text{proof} \rangle$

end

46 Polynomial Division over a Field

context *Field*
begin

46.1 Units of $F[X]$

Over a field, multiplication of nonzero polynomials adds degrees, so a polynomial can only be a multiplicative unit if it has degree 0: the units of $F[X]$ are exactly the nonzero constants. This pins down what it means for an irreducible polynomial to admit only trivial factorisations.

A nonzero constant is a unit, with the constant of the inverse coefficient as inverse.

lemma *poly-const-unit:*
assumes $a: a \in R$ **and** $anz: a \neq \mathbf{0}$
shows *poly-unit* (*poly-const* a)
 $\langle \text{proof} \rangle$

Conversely, every unit of $F[X]$ is a nonzero constant. An invertible polynomial is nonzero and has degree 0 (degrees add under multiplication), so it is the constant of its own (nonzero) leading coefficient.

lemma *poly-unit-imp-const*:
assumes *inv*: *poly-unit* *p*
shows $p \neq \mathbf{0}_P \wedge \text{degree } p = 0 \wedge p = \text{poly-const } (p \ 0) \wedge p \ 0 \neq \mathbf{0}$
 $\langle \text{proof} \rangle$

A polynomial is *irreducible* when it is a non-unit, nonzero, and every factorisation has a unit factor. Over a field this excludes the units (the nonzero constants) and the zero polynomial, matching the usual notion for $F[X]$.

definition *poly-irreducible* :: $(\text{nat} \Rightarrow 'a) \Rightarrow \text{bool}$
where *poly-irreducible* *p* $\equiv$
 $p \in \text{poly-carrier} \wedge p \neq \mathbf{0}_P \wedge \neg \text{poly-unit } p$
 $\wedge (\forall a \ b. a \in \text{poly-carrier} \longrightarrow b \in \text{poly-carrier} \longrightarrow p = a \otimes_P b$
 $\longrightarrow \text{poly-unit } a \vee \text{poly-unit } b)$

lemma *poly-irreducibleD-carrier*: *poly-irreducible* *p* $\implies p \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

lemma *poly-irreducibleD-nonzero*: *poly-irreducible* *p* $\implies p \neq \mathbf{0}_P$
 $\langle \text{proof} \rangle$

lemma *poly-irreducibleD-nonunit*: *poly-irreducible* *p* $\implies \neg \text{poly-unit } p$
 $\langle \text{proof} \rangle$

lemma *poly-irreducibleD-factor*:
 $\llbracket \text{poly-irreducible } p; a \in \text{poly-carrier}; b \in \text{poly-carrier}; p = a \otimes_P b \rrbracket$
 $\implies \text{poly-unit } a \vee \text{poly-unit } b$
 $\langle \text{proof} \rangle$

An irreducible polynomial is nonconstant: a nonzero constant is a unit, and the zero polynomial is excluded outright. Needed because the remainders modulo *p* contain $\mathbf{1}_P$ only when *p* is nonconstant.

lemma *poly-irreducible-degree-pos*:
assumes *irr*: *poly-irreducible* *p* **shows** $0 < \text{degree } p$
 $\langle \text{proof} \rangle$

interpretation *Ring-R*: *Ring R addition* $(\cdot) \ \mathbf{0} \ \mathbf{1} \ \langle \text{proof} \rangle$

The reduction step of the division algorithm: when $\text{degree } b \leq \text{degree } a$ and both are nonzero, subtracting the appropriate monomial multiple of *b* cancels the leading term of *a*, strictly decreasing its degree.

lemma *div-step*:
assumes *a*: $a \in \text{poly-carrier}$ **and** *b*: $b \in \text{poly-carrier}$
and *anz*: $a \neq \mathbf{0}_P$ **and** *bnz*: $b \neq \mathbf{0}_P$ **and** *ge*: $\text{degree } b \leq \text{degree } a$
obtains *m* **where** $m \in \text{poly-carrier}$ $\text{degree } m = \text{degree } a - \text{degree } b$
and $a \oplus_P (\ominus_P (m \otimes_P b)) = \mathbf{0}_P \vee \text{degree } (a \oplus_P (\ominus_P (m \otimes_P b))) < \text{degree } a$
 $\langle \text{proof} \rangle$

The division algorithm. For any dividend *a* and nonzero divisor *b* there exist a quotient *q* and remainder *r* with $a = b \otimes q \oplus r$ and the remainder either zero

or of degree below that of b . Proved by strong induction on *degree a*, repeatedly applying *div-step*.

theorem *poly-divide*:

assumes $a: a \in \text{poly-carrier}$ **and** $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
shows $\exists q \ r. q \in \text{poly-carrier} \wedge r \in \text{poly-carrier} \wedge a = (b \otimes_P q) \oplus_P r$
 $\wedge (r = \mathbf{0}_P \vee \text{degree } r < \text{degree } b)$
 $\langle \text{proof} \rangle$

Uniqueness of quotient and remainder. If b is nonzero and two quotient/remainder pairs both work — each remainder of degree below b — then they coincide. For if the quotients differed, $b \otimes (q_1 \oplus \ominus q_2)$ would have degree $\geq \text{degree } b$ (degrees add over a domain), yet it equals $r_2 \oplus \ominus r_1$, of degree below b .

Left cancellation of a nonzero factor: $a \otimes x = a \otimes y \implies x = y$ (the ring of polynomials over a field is a domain).

lemma *poly-mult-left-cancel*:

assumes $a: a \in \text{poly-carrier}$ **and** $\text{anz}: a \neq \mathbf{0}_P$
and $x: x \in \text{poly-carrier}$ **and** $y: y \in \text{poly-carrier}$
and $\text{eq}: a \otimes_P x = a \otimes_P y$
shows $x = y$
 $\langle \text{proof} \rangle$

theorem *poly-divide-unique*:

assumes $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
and $q1: q1 \in \text{poly-carrier}$ **and** $r1: r1 \in \text{poly-carrier}$
and $q2: q2 \in \text{poly-carrier}$ **and** $r2: r2 \in \text{poly-carrier}$
and $e1: a = (b \otimes_P q1) \oplus_P r1$ **and** $d1: r1 = \mathbf{0}_P \vee \text{degree } r1 < \text{degree } b$
and $e2: a = (b \otimes_P q2) \oplus_P r2$ **and** $d2: r2 = \mathbf{0}_P \vee \text{degree } r2 < \text{degree } b$
shows $q1 = q2 \wedge r1 = r2$
 $\langle \text{proof} \rangle$

46.2 Remainder modulo a polynomial

The remainder of a on division by b : the canonical representative of a modulo b , of degree below that of b . Defined by choice from the division algorithm, and pinned down uniquely by *poly-divide-unique*.

definition $\text{pmod} :: (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a)$

where $\text{pmod } a \ b = (\text{SOME } r. \exists q. q \in \text{poly-carrier} \wedge r \in \text{poly-carrier}$
 $\wedge a = (b \otimes_P q) \oplus_P r \wedge (r = \mathbf{0}_P \vee \text{degree } r < \text{degree } b))$

The defining property of *pmod*: there is a quotient realising it.

lemma *pmod-spec*:

assumes $a: a \in \text{poly-carrier}$ **and** $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
shows $\exists q. q \in \text{poly-carrier} \wedge \text{pmod } a \ b \in \text{poly-carrier}$
 $\wedge a = (b \otimes_P q) \oplus_P \text{pmod } a \ b \wedge (\text{pmod } a \ b = \mathbf{0}_P \vee \text{degree } (\text{pmod } a \ b) <$
 $\text{degree } b)$
 $\langle \text{proof} \rangle$

lemma *pmod-closed*:

$\llbracket a \in \text{poly-carrier}; b \in \text{poly-carrier}; b \neq \mathbf{0}_P \rrbracket \implies \text{pmod } a \ b \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

lemma *pmod-degree*:

$\llbracket a \in \text{poly-carrier}; b \in \text{poly-carrier}; b \neq \mathbf{0}_P \rrbracket$
 $\implies \text{pmod } a \ b = \mathbf{0}_P \vee \text{degree } (\text{pmod } a \ b) < \text{degree } b$
 $\langle \text{proof} \rangle$

pmod is the unique remainder: any low-degree r congruent to a modulo b equals $\text{pmod } a \ b$.

lemma *pmod-unique*:

assumes $a: a \in \text{poly-carrier}$ **and** $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
and $q: q \in \text{poly-carrier}$ **and** $r: r \in \text{poly-carrier}$
and $\text{eq}: a = (b \otimes_P q) \oplus_P r$ **and** $\text{rd}: r = \mathbf{0}_P \vee \text{degree } r < \text{degree } b$
shows $\text{pmod } a \ b = r$
 $\langle \text{proof} \rangle$

A remainder is a low-degree polynomial — the form in which the remainders are used as a carrier.

lemma *pmod-low-poly*:

assumes $a: a \in \text{poly-carrier}$ **and** $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
shows $\text{pmod } a \ b \in \text{low-poly } (\text{degree } b)$
 $\langle \text{proof} \rangle$

A polynomial of degree below b is its own remainder.

lemma *pmod-small*:

assumes $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
and $r: r \in \text{poly-carrier}$ **and** $\text{rd}: r = \mathbf{0}_P \vee \text{degree } r < \text{degree } b$
shows $\text{pmod } r \ b = r$
 $\langle \text{proof} \rangle$

Changing a by a multiple of b does not change the remainder. Everything below is a consequence of this and *pmod-unique*.

lemma *pmod-shift*:

assumes $y: y \in \text{poly-carrier}$ **and** $c: c \in \text{poly-carrier}$
and $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
shows $\text{pmod } ((b \otimes_P c) \oplus_P y) \ b = \text{pmod } y \ b$
 $\langle \text{proof} \rangle$

The remainder is additive outright: a sum of two low-degree remainders is already low-degree, so no second reduction is needed.

lemma *pmod-add*:

assumes $a: a \in \text{poly-carrier}$ **and** $a': a' \in \text{poly-carrier}$
and $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
shows $\text{pmod } (a \oplus_P a') \ b = \text{pmod } a \ b \oplus_P \text{pmod } a' \ b$
 $\langle \text{proof} \rangle$

The remainder is multiplicative *after a further reduction*: a product of two low-degree remainders need not be low-degree, so the outer *pmod* cannot be dropped here as it was for the sum. This is the congruence that makes the remainders a ring.

lemma *pmod-mult*:

assumes $a: a \in \text{poly-carrier}$ **and** $a': a' \in \text{poly-carrier}$
and $b: b \in \text{poly-carrier}$ **and** $\text{bnz}: b \neq \mathbf{0}_P$
shows $\text{pmod } (a \otimes_P a') b = \text{pmod } (\text{pmod } a b \otimes_P \text{pmod } a' b) b$
 $\langle \text{proof} \rangle$

46.3 The factor theorem

The root factor *root-factor* a has degree exactly 1: its degree-1 coefficient is $\mathbf{1}$, which is nonzero in a (nontrivial) field.

lemma *degree-root-factor*:

assumes $a: a \in R$ **shows** $\text{degree } (\text{root-factor } a) = 1$
 $\langle \text{proof} \rangle$

lemma *root-factor-nonzero*:

assumes $a: a \in R$ **shows** $\text{root-factor } a \neq \mathbf{0}_P$
 $\langle \text{proof} \rangle$

The factor theorem. A field element a is a root of p iff the linear polynomial *root-factor* a divides p .

theorem *factor-theorem*:

assumes $p: p \in \text{poly-carrier}$ **and** $a: a \in R$
shows $\text{eval } a p = \mathbf{0} \iff (\exists q. q \in \text{poly-carrier} \wedge p = \text{root-factor } a \otimes_P q)$
 $\langle \text{proof} \rangle$

46.4 A nonzero polynomial has at most *degree* roots

Evaluation of the linear factor *root-factor* a at an arbitrary point b is $b - a$; in particular it is nonzero unless $b = a$.

lemma *eval-root-factor-apply*:

assumes $a: a \in R$ **and** $b: b \in R$ **shows** $\text{eval } b (\text{root-factor } a) = b - a$
 $\langle \text{proof} \rangle$

The root set $\{a \in R. \text{eval } a p = \mathbf{0}\}$ of a nonzero polynomial over a field is finite and has at most *degree* p elements. By induction on the degree: a root a lets the factor theorem write $p = \text{root-factor } a \otimes_P q$ with $\text{degree } q = \text{degree } p - 1$ (*degree-mult*, *degree-root-factor*); as a field has no zero divisors, every root of p is a or a root of q .

theorem *card-roots-le-degree*:

$p \in \text{poly-carrier} \implies p \neq \mathbf{0}_P$
 $\implies \text{finite } \{a \in R. \text{eval } a p = \mathbf{0}\} \wedge \text{card } \{a \in R. \text{eval } a p = \mathbf{0}\} \leq \text{degree } p$
 $\langle \text{proof} \rangle$

46.5 Low-degree irreducibility

Evaluation of a polynomial of degree ≤ 1 : $eval\ \alpha\ c = c\ 0 + c\ 1 \cdot \alpha$.

lemma *eval-le1*:

assumes $c: c \in poly\text{-}carrier$ **and** $al: \alpha \in R$ **and** $d: degree\ c \leq 1$
shows $eval\ \alpha\ c = c\ 0 + c\ 1 \cdot \alpha$

<proof>

Every polynomial of degree 1 over a field has a root: $-(c\ 0) \cdot (c\ 1)^{-1}$.

lemma *degree1-has-root*:

assumes $c: c \in poly\text{-}carrier$ **and** $d: degree\ c = 1$
shows $\exists \alpha \in R. eval\ \alpha\ c = 0$

<proof>

lemma *degree-nz*:

assumes $a: a \in poly\text{-}carrier$ **and** $nua: \neg poly\text{-}unit\ a$ **and** $anz: a \neq 0_P$
shows $degree\ a \neq 0$

<proof>

A degree-1 factor of p supplies a root of p : if $p = a \otimes_P b$ with a (or b) of degree 1, evaluation at the factor's root kills p .

lemma *degree1-factor-root*:

assumes $a: a \in poly\text{-}carrier$ **and** $b \in poly\text{-}carrier$ $p = a \otimes_P b$ $degree\ a = 1$
shows $\exists \alpha \in R. eval\ \alpha\ p = 0$

<proof>

Degree-2, 3 irreducibility criterion. A cubic with no root is irreducible: a non-trivial factorisation $p = a \otimes_P b$ has $degree\ a + degree\ b = 2$ or 3 with both factors of degree ≥ 1 , so one of them has degree 1 and hence a root of p .

theorem *degree23-no-root-irreducible*:

assumes $p: p \in poly\text{-}carrier$ **and** $d: degree\ p \in \{2, 3\}$
and $noroot: \bigwedge \alpha. \alpha \in R \implies eval\ \alpha\ p \neq 0$
shows $poly\text{-}irreducible\ p$

<proof>

46.6 Factorisation into irreducibles

The product of a list of polynomials (right fold, empty list giving 1_P).

definition $poly\text{-}prod :: (nat \Rightarrow 'a) list \Rightarrow (nat \Rightarrow 'a)$

where $poly\text{-}prod\ fs \equiv foldr\ (\otimes_P)\ fs\ 1_P$

lemma *poly-prod-Nil [simp]*: $poly\text{-}prod\ [] = 1_P$

<proof>

lemma *poly-prod-Cons [simp]*: $poly\text{-}prod\ (f \# fs) = f \otimes_P poly\text{-}prod\ fs$

<proof>

lemma *poly-prod-closed*:

$(\bigwedge f. f \in \text{set } fs \implies f \in \text{poly-carrier}) \implies \text{poly-prod } fs \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

lemma *poly-prod-append*:

assumes $fs: \bigwedge f. f \in \text{set } fs \implies f \in \text{poly-carrier}$
and $gs: \bigwedge g. g \in \text{set } gs \implies g \in \text{poly-carrier}$
shows $\text{poly-prod } (fs @ gs) = \text{poly-prod } fs \otimes_P \text{poly-prod } gs$
 $\langle \text{proof} \rangle$

46.7 Splitting into linear factors

A polynomial *splits* when it is a constant times a product of root factors, all of whose roots lie in the field. Equivalently it is a product of linear factors: the constant carries the leading coefficient. Taking that constant to range over the whole field, rather than the nonzero elements, lets $\mathbf{0}_P$ split as well (empty list, zero constant), so no polynomial is excluded.

definition *splits* :: $(\text{nat} \Rightarrow 'a) \Rightarrow \text{bool}$

where $\text{splits } p \equiv (\exists c \text{ as. } c \in R \wedge \text{set as} \subseteq R$
 $\wedge p = \text{poly-const } c \otimes_P \text{poly-prod } (\text{List.map root-factor as}))$

lemma *splitsI*:

$\llbracket c \in R; \text{set as} \subseteq R; p = \text{poly-const } c \otimes_P \text{poly-prod } (\text{List.map root-factor as}) \rrbracket$
 $\implies \text{splits } p$
 $\langle \text{proof} \rangle$

lemma *splitsE*:

assumes *splits* p
obtains $c \text{ as}$ **where** $c \in R \text{ set as} \subseteq R$
and $p = \text{poly-const } c \otimes_P \text{poly-prod } (\text{List.map root-factor as})$
 $\langle \text{proof} \rangle$

lemma *splits-closed*: $\text{splits } p \implies p \in \text{poly-carrier}$

$\langle \text{proof} \rangle$

lemma *splits-const*:

assumes $c: c \in R$ **shows** *splits* $(\text{poly-const } c)$
 $\langle \text{proof} \rangle$

lemma *splits-zero* [*simp*]: *splits* $\mathbf{0}_P$

$\langle \text{proof} \rangle$

lemma *splits-root-factor*:

assumes $a: a \in R$ **shows** *splits* $(\text{root-factor } a)$
 $\langle \text{proof} \rangle$

Splitting is multiplicative: concatenate the root lists and multiply the constants.

lemma *splits-mult*:

assumes p : *splits* p **and** q : *splits* q **shows** *splits* $(p \otimes_P q)$
 ⟨*proof*⟩

A polynomial of degree one splits. Divide by the leading coefficient: the result is *root-factor* a for a the negated ratio of the two coefficients. This is the base case that makes a factorisation into linear factors out of a factorisation into irreducibles all of whose factors have degree one.

lemma *degree-one-splits*:

assumes q : $q \in \text{poly-carrier}$ **and** d : *degree* $q = 1$
shows *splits* q
 ⟨*proof*⟩

lemma *splits-poly-prod*:

assumes $\bigwedge f. f \in \text{set } fs \implies \text{splits } f$ **shows** *splits* $(\text{poly-prod } fs)$
 ⟨*proof*⟩

Existence of an irreducible factorisation. Every nonzero non-unit polynomial over a field is a product of a nonempty list of irreducibles. By strong induction on the degree: an irreducible p is its own singleton factorisation; otherwise $p = a \otimes_P b$ with both factors non-units — hence, over a field, of strictly smaller degree — and the induction hypothesis factors each.

theorem *factorization-into-irreducibles*:

assumes $p \in \text{poly-carrier}$ **and** $p \neq \mathbf{0}_P$ **and** $\neg \text{poly-unit } p$
shows $\exists fs. fs \neq [] \wedge (\forall f \in \text{set } fs. \text{poly-irreducible } f) \wedge p = \text{poly-prod } fs$
 ⟨*proof*⟩

46.8 Monic polynomials and normalisation

A polynomial is *monic* when it is nonzero with leading coefficient $\mathbf{1}$.

definition *poly-monic* :: $(\text{nat} \Rightarrow 'a) \Rightarrow \text{bool}$
where *poly-monic* $p \equiv p \neq \mathbf{0}_P \wedge \text{lead-coeff } p = \mathbf{1}$

Scaling a nonzero polynomial by a nonzero constant preserves its degree and scales its leading coefficient.

lemma *degree-poly-const-mult*:

assumes $c: c \in R$ **and** $\text{cnz}: c \neq \mathbf{0}$ **and** $p: p \in \text{poly-carrier}$ **and** $\text{pnz}: p \neq \mathbf{0}_P$
shows *degree* $(\text{poly-const } c \otimes_P p) = \text{degree } p$
 ⟨*proof*⟩

lemma *lead-coeff-poly-const-mult*:

assumes $c: c \in R$ **and** $\text{cnz}: c \neq \mathbf{0}$ **and** $p: p \in \text{poly-carrier}$ **and** $\text{pnz}: p \neq \mathbf{0}_P$
shows *lead-coeff* $(\text{poly-const } c \otimes_P p) = c \cdot \text{lead-coeff } p$
 ⟨*proof*⟩

Every nonzero polynomial has a monic associate, obtained by scaling by the inverse of its leading coefficient.

theorem *monic-associate-exists*:

assumes $p: p \in \text{poly-carrier}$ **and** $\text{pnz}: p \neq \mathbf{0}_P$
shows $\exists u m. u \in R \wedge u \neq \mathbf{0} \wedge m \in \text{poly-carrier} \wedge \text{poly-monic } m \wedge p = \text{poly-const}$
 $u \otimes_P m$
 $\langle \text{proof} \rangle$

46.9 The polynomial ring over a field is a principal ideal domain

Every ideal of $F[X]$ is principal. Given an ideal J , the zero ideal is generated by $\mathbf{0}_P$; otherwise pick a nonzero b of *least degree* in J . Dividing any $a \in J$ by b (the division algorithm *poly-divide*) leaves a remainder that again lies in J but has degree below that of b , so by minimality it must be zero: hence a is a multiple of b . This is the structural fact behind Kronecker's construction — it makes (p) maximal for irreducible p , and so $F[X]/(p)$ a field.

theorem *poly-PID:*

assumes J : Ideal J *poly-carrier* $(\oplus_P) (\otimes_P) \mathbf{0}_P \mathbf{1}_P$
shows $\exists b. b \in \text{poly-carrier} \wedge J = \{r \otimes_P b \mid r. r \in \text{poly-carrier}\}$
 $\langle \text{proof} \rangle$

end

47 Pushing a Polynomial along a Ring Homomorphism

Coefficientwise application of a ring homomorphism commutes with every operation of the polynomial ring..

context

fixes $A :: 'b \text{ set}$ **and** $\text{add mult} :: 'b \Rightarrow 'b \Rightarrow 'b$ **and** $z u :: 'b$
and $B :: 'c \text{ set}$ **and** $\text{badd bmult} :: 'c \Rightarrow 'c \Rightarrow 'c$ **and** $\text{bz bu} :: 'c$
and $h :: 'b \Rightarrow 'c$
assumes RA : Ring A $\text{add mult } z u$ **and** RB : Ring B $\text{badd bmult } bz bu$
and hB : $\bigwedge y. y \in A \implies h y \in B$
and hz : $h z = bz$ **and** hu : $h u = bu$
and $hadd$: $\bigwedge y y'. \llbracket y \in A; y' \in A \rrbracket \implies h (\text{add } y y') = \text{badd } (h y) (h y')$
and $hmult$: $\bigwedge y y'. \llbracket y \in A; y' \in A \rrbracket \implies h (\text{mult } y y') = \text{bmult } (h y) (h y')$

begin

lemma cmA : commutative-monoid A $\text{add } z$
 $\langle \text{proof} \rangle$

lemma cmB : commutative-monoid B $\text{badd } bz$
 $\langle \text{proof} \rangle$

lemma gpA : Abelian-Group A $\text{add } z$
 $\langle \text{proof} \rangle$

lemma *gpB*: Abelian-Group B badd bz
 $\langle proof \rangle$

lemma *monA*: Monoid A mult u
 $\langle proof \rangle$

lemma *poly-zero-hom*: $(\lambda k. h (Ring.poly-zero\ z\ k)) = Ring.poly-zero\ bz$
 $\langle proof \rangle$

lemma *poly-one-hom*: $(\lambda k. h (Ring.poly-one\ z\ u\ k)) = Ring.poly-one\ bz\ bu$
 $\langle proof \rangle$

lemma *poly-const-hom*: $(\lambda k. h (Ring.poly-const\ z\ c\ k)) = Ring.poly-const\ bz\ (h\ c)$
 $\langle proof \rangle$

lemma *var-hom*: $(\lambda k. h (Ring.var\ z\ u\ k)) = Ring.var\ bz\ bu$
 $\langle proof \rangle$

Coefficients of the polynomials named above, needed as closure side conditions below.

lemma *zA*: $z \in A$
 $\langle proof \rangle$

lemma *uA*: $u \in A$
 $\langle proof \rangle$

lemma *poly-const-coeff-closed*: $c \in A \implies Ring.poly-const\ z\ c\ k \in A$
 $\langle proof \rangle$

lemma *var-coeff-closed*: $Ring.var\ z\ u\ k \in A$
 $\langle proof \rangle$

lemma *poly-add-hom*:
assumes $p: \bigwedge k. p\ k \in A$ **and** $q: \bigwedge k. q\ k \in A$
shows $(\lambda k. h (Ring.poly-add\ add\ p\ q\ k)) = Ring.poly-add\ badd\ (\lambda k. h\ (p\ k))\ (\lambda k. h\ (q\ k))$
 $\langle proof \rangle$

The additive inverse is pinned by the equation it solves, so a homomorphism of the additive groups carries it along.

lemma *inverse-hom*:
assumes $y \in A$
shows $h (Monoid.inverse\ A\ add\ z\ y) = Monoid.inverse\ B\ badd\ bz\ (h\ y)$
 $\langle proof \rangle$

lemma *poly-neg-hom*:
assumes $p: \bigwedge k. p\ k \in A$
shows $(\lambda k. h (Ring.poly-neg\ A\ add\ z\ p\ k)) = Ring.poly-neg\ B\ badd\ bz\ (\lambda k. h\ (p\ k))$
 $\langle proof \rangle$

lemma *poly-neg-coeff-closed*:
assumes $p: \bigwedge k. p\ k \in A$
shows $\text{Ring.poly-neg } A \text{ add } z\ p\ k \in A$
 $\langle \text{proof} \rangle$

Convolution is a finite composite of products, so it pushes along by *fincomp-hom*. Only the two monoid premises are supplied: resolving the homomorphism premise inside *OF* sends unification into higher-order search, as the corresponding step of *eval-hom* also avoids.

lemma *poly-mult-hom*:
assumes $p: \bigwedge k. p\ k \in A$ **and** $q: \bigwedge k. q\ k \in A$
shows $(\lambda k. h\ (\text{Ring.poly-mult } A \text{ add mult } z\ p\ q\ k))$
 $= \text{Ring.poly-mult } B \text{ badd bmult } bz\ (\lambda k. h\ (p\ k))\ (\lambda k. h\ (q\ k))$
 $\langle \text{proof} \rangle$

lemma *poly-mult-coeff-closed*:
assumes $p: \bigwedge k. p\ k \in A$ **and** $q: \bigwedge k. q\ k \in A$
shows $\text{Ring.poly-mult } A \text{ add mult } z\ p\ q\ k \in A$
 $\langle \text{proof} \rangle$

lemma *poly-one-coeff-closed*: $\text{Ring.poly-one } z\ u\ k \in A$
 $\langle \text{proof} \rangle$

The linear factor $X - a$, the list product, and hence the property of splitting. The stronger structure is taken as an ordinary assumption of each lemma, so that the ring-level facts above keep their generality.

lemma *root-factor-coeff-closed*:
assumes $\text{commutative-ring } A \text{ add mult } z\ u\ a \in A$
shows $\text{commutative-ring.root-factor } A \text{ add } z\ u\ a\ k \in A$
 $\langle \text{proof} \rangle$

lemma *root-factor-hom*:
assumes $CA: \text{commutative-ring } A \text{ add mult } z\ u$ **and** $CB: \text{commutative-ring } B \text{ badd bmult } bz\ bu$
and $a: a \in A$
shows $(\lambda k. h\ (\text{commutative-ring.root-factor } A \text{ add } z\ u\ a\ k)) = \text{commutative-ring.root-factor } B \text{ badd } bz\ bu\ (h\ a)$
 $\langle \text{proof} \rangle$

lemma *poly-prod-coeff-closed*:
assumes $FA: \text{Field } A \text{ add mult } z\ u$
and $fs: \bigwedge f. f \in \text{set } fs \implies (\forall k. f\ k \in A)$
shows $\text{Field.poly-prod } A \text{ add mult } z\ u\ fs\ k \in A$
 $\langle \text{proof} \rangle$

lemma *poly-prod-hom*:
assumes $FA: \text{Field } A \text{ add mult } z\ u$ **and** $FB: \text{Field } B \text{ badd bmult } bz\ bu$
and $fs: \bigwedge f. f \in \text{set } fs \implies (\forall k. f\ k \in A)$

```

shows ( $\lambda k. h (Field.poly-prod A add mult z u fs k)$ )
  =  $Field.poly-prod B badd bmult bz bu (List.map (\lambda f k. h (f k)) fs)$ 
 $\langle proof \rangle$ 

```

Splitting is preserved: the witnessing constant and root list are simply carried across. The steps use *simp only* throughout, to keep *map-map* from rewriting the two lists into composed form and so out of reach of the transport equations.

theorem *splits-hom*:

```

assumes  $FA: Field A add mult z u$  and  $FB: Field B badd bmult bz bu$ 
and  $sp: Field.splits A add mult z u p$ 
shows  $Field.splits B badd bmult bz bu (\lambda k. h (p k))$ 
 $\langle proof \rangle$ 

```

end

end

48 Prime and Maximal Ideals

theory *Ideal-Theory*

imports *Ring-Theory HOL.Zorn*

begin

Phase B: prime and maximal ideals, and the quotient characterisations.

context *Ideal*

begin

Prime and maximal ideals: proper ideals with the usual closure / maximality property.

definition *prime-ideal* :: *bool*

where $prime-ideal \longleftrightarrow I \neq R \wedge (\forall a \in R. \forall b \in R. a \cdot b \in I \longrightarrow a \in I \vee b \in I)$

definition *maximal-ideal* :: *bool*

where $maximal-ideal \longleftrightarrow I \neq R \wedge (\forall J. Ideal J R (+) (\cdot) \mathbf{0} \mathbf{1} \longrightarrow I \subseteq J \longrightarrow J = I \vee J = R)$

The coset of an element and the set of all cosets (the quotient R/I), given clean names so they can be referred to from interpretations at the top level (the underlying *additive.Class/additive.Partition* live behind several sublocale hops and have no stable qualified name).

definition *coset* :: $'a \Rightarrow 'a \text{ set}$

where $coset a = additive.Class a$

definition *quotient-set* :: $'a \text{ set set}$

where $quotient-set = additive.Partition$

lemma *coset-eq-iff*:

assumes $a \in R \ b \in R$ **shows** $\text{coset } a = \text{coset } b \longleftrightarrow a + (-b) \in I$
 $\langle \text{proof} \rangle$

lemma *coset-in-quotient-set*: $a \in R \implies \text{coset } a \in \text{quotient-set}$
 $\langle \text{proof} \rangle$

lemma *quotient-set-repr*: $C \in \text{quotient-set} \implies \exists a \in R. C = \text{coset } a$
 $\langle \text{proof} \rangle$

Membership criterion: a coset is the zero coset iff its representative is in the ideal.

lemma *Class-eq-zero-iff*:
assumes $a \in R$
shows $\text{additive.Class } a = \text{additive.Class } \mathbf{0} \longleftrightarrow a \in I$
 $\langle \text{proof} \rangle$

end

An ideal of a commutative ring: add only the commutativity axiom on top of *Ideal* (so the *Ring/multiplicative* structure is shared, avoiding a locale-merge clash).

locale *ideal-in-comm-ring* = *Ideal* + *commutative-ring*
begin

lemmas *comm* = *multiplicative.commutative*

The quotient multiplication is commutative.

lemma *quot-mult-comm*:
assumes $a \in R \ b \in R$
shows $\text{additive.Class } a \ [\cdot] \ \text{additive.Class } b = \text{additive.Class } b \ [\cdot] \ \text{additive.Class } a$
 $\langle \text{proof} \rangle$

Quotient multiplication in terms of representatives.

lemma *quot-mult-Class*:
assumes $a \in R \ b \in R$
shows $\text{additive.Class } a \ [\cdot] \ \text{additive.Class } b = \text{additive.Class } (a \cdot b)$
 $\langle \text{proof} \rangle$

B1: the ideal is prime iff the quotient has no zero divisors (is an integral domain, given nontriviality $I \neq R$).

theorem *prime-ideal-iff-quotient-no-zero-divisors*:
 $\text{prime-ideal} \longleftrightarrow I \neq R \wedge$
 $(\forall a \in R. \forall b \in R. \text{additive.Class } a \ [\cdot] \ \text{additive.Class } b = \text{additive.Class } \mathbf{0}$
 $\longrightarrow \text{additive.Class } a = \text{additive.Class } \mathbf{0} \vee \text{additive.Class } b = \text{additive.Class } \mathbf{0})$
 $\langle \text{proof} \rangle$

The ideal generated by I together with one element a , i.e. $I + (a)$.

definition *ext-ideal* :: 'a $\Rightarrow$ 'a set
where *ext-ideal* a = {m + r · a | m r. m ∈ I ∧ r ∈ R}

lemma *ext-ideal-memI*: $\llbracket m \in I; r \in R \rrbracket \Rightarrow m + r \cdot a \in \text{ext-ideal } a$
 ⟨proof⟩

lemma *ext-ideal-contains-I*:
assumes a ∈ R
shows I ⊆ *ext-ideal* a
 ⟨proof⟩

lemma *ext-ideal-contains-elt*:
assumes a ∈ R
shows a ∈ *ext-ideal* a
 ⟨proof⟩

ext-ideal a is closed under addition and additive inverses, hence an additive subgroup of R.

lemma *ext-ideal-subgroup*:
assumes a: a ∈ R
shows Subgroup (*ext-ideal* a) R (+) 0
 ⟨proof⟩

Hence *ext-ideal* a is an ideal of R: the absorption laws follow from I's absorption and commutativity of multiplication.

lemma *ext-ideal-is-ideal*:
assumes a: a ∈ R
shows Ideal (*ext-ideal* a) R (+) (·) 0 1
 ⟨proof⟩

For a maximal ideal, any element outside I generates (with I) the whole ring, giving a representation $\mathbf{1} = m + r \cdot a$ with m ∈ I.

lemma *maximal-ext-ideal-eq-whole*:
assumes max: maximal-ideal **and** a: a ∈ R **and** aI: a ∉ I
shows *ext-ideal* a = R
 ⟨proof⟩

A maximal ideal is prime. If a · b ∈ I but a ∉ I, then adjoining a generates the whole ring, giving $\mathbf{1} = m + r \cdot a$ with m ∈ I; hence $b = m \cdot b + r \cdot (a \cdot b) \in I$.

theorem *maximal-imp-prime-ideal*:
assumes max: maximal-ideal **shows** prime-ideal
 ⟨proof⟩

lemma *maximal-quotient-invertible*:
assumes max: maximal-ideal **and** a: a ∈ R **and** aI: a ∉ I
shows $\exists r \in R. \text{additive.Class } r \text{ } [\cdot] \text{ additive.Class } a = \text{additive.Class } \mathbf{1}$
 ⟨proof⟩

If two ring elements have equal cosets, their difference lies in I .

lemma *Class-eq-imp-diff-in-I:*

assumes $a \in R \ b \in R$ *additive.Class* $a = \text{additive.Class } b$

shows $a - b \in I$

<proof>

theorem *maximal-ideal-iff-quotient-field:*

maximal-ideal $\longleftrightarrow I \neq R \wedge$

$(\forall a \in R. \text{additive.Class } a \neq \text{additive.Class } \mathbf{0})$

$\longrightarrow (\exists r \in R. \text{additive.Class } r [\cdot] \text{additive.Class } a = \text{additive.Class } \mathbf{1}))$

<proof>

end

49 Principal Ideals

The principal ideal generated by a single element of a commutative ring: the set of all multiples $r \cdot a$. This is the smallest ideal containing a , and it is the building block for principal ideal domains (e.g. $F[X]$ over a field).

49.1 Existence of maximal ideals

Every proper ideal is contained in a maximal one. This is the standard application of Zorn's lemma (*Zorn-Lemma2*) to the family of proper ideals containing a given one: the union of a chain of such ideals is again one, so the family has a maximal element.

The only delicate point is *properness* of the union. An ideal is proper exactly when it omits the unit $\mathbf{1}$: if $\mathbf{1}$ lies in an ideal then so does $a \cdot \mathbf{1} = a$ for every a , making it the whole ring. Since every member of the chain omits $\mathbf{1}$, so does the union — which is why the chain condition can be checked at all, and why the argument needs a ring with a unit.

context *Ring*

begin

An ideal is proper if and only if it does not contain $\mathbf{1}$.

lemma *ideal-proper-iff-one-notin:*

assumes $I: \text{Ideal } I \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

shows $I \neq R \longleftrightarrow \mathbf{1} \notin I$

<proof>

The union of a nonempty chain of ideals is an ideal. Directedness is what makes the additive closure work: two elements of the union lie in a common member of the chain.

lemma *ideal-Union-chain:*

assumes $ne: \mathcal{C} \neq \{\}$

and *id*: $\bigwedge I. I \in \mathcal{C} \implies \text{Ideal } I \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$
and *chain*: $\bigwedge I J. \llbracket I \in \mathcal{C}; J \in \mathcal{C} \rrbracket \implies I \subseteq J \vee J \subseteq I$
shows *Ideal* $(\bigcup \mathcal{C}) \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$
<proof>

Krull's theorem. Every proper ideal extends to a maximal one. Apply *Zorn-Lemma2* to the family of proper ideals containing I : a nonempty chain's union is an ideal by *ideal-Union-chain*, is proper because no member contains $\mathbf{1}$, and contains I ; the empty chain is bounded by I itself.

theorem *maximal-ideal-exists*:

assumes I : *Ideal* $I \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$ **and** *proper*: $I \neq R$
shows $\exists M. \text{Ideal } M \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1} \wedge I \subseteq M \wedge \text{Ideal.maximal-ideal } M \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$
<proof>

end

context *commutative-ring*

begin

definition *principal-ideal* :: $'a \Rightarrow 'a \text{ set}$

where *principal-ideal* $a = \{r \cdot a \mid r. r \in R\}$

lemma *principal-ideal-mem*: $r \in R \implies r \cdot a \in \text{principal-ideal } a$
<proof>

lemma *principal-ideal-subset*:

assumes $a \in R$ **shows** *principal-ideal* $a \subseteq R$
<proof>

a itself lies in *principal-ideal* a (take $r = \mathbf{1}$).

lemma *principal-ideal-contains*:

assumes $a \in R$ **shows** $a \in \text{principal-ideal } a$
<proof>

principal-ideal a is an additive subgroup: closed under $+$ via distributivity, and under inverses via *left-minus*.

lemma *principal-ideal-subgroup*:

assumes $a: a \in R$ **shows** *Subgroup* $(\text{principal-ideal } a) \text{ } R \text{ } (+) \text{ } \mathbf{0}$
<proof>

Hence *principal-ideal* a is an ideal: the absorption laws follow from associativity and commutativity of multiplication.

lemma *principal-ideal-is-ideal*:

assumes $a: a \in R$ **shows** *Ideal* $(\text{principal-ideal } a) \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$
<proof>

end

end

50 Ideals of the Polynomial Ring; Kronecker's Construction

theory *Poly-Ideal*

imports *Poly-Ring Ideal-Theory HOL-Library.Multiset*

begin

This theory joins the carrier-set polynomial ring *Poly-Ring* to the ideal theory of *Ideal-Theory*. The payoff is Kronecker's lemma: over a field, the ideal generated by an irreducible polynomial is maximal, so the quotient $F[X]/(p)$ is a field. This is the engine that builds field extensions (and, ultimately, finite fields) by adjoining roots of irreducible polynomials.

context *Field*

begin

Inside a field the polynomial ring is a commutative ring, so its principal ideals are available through *commutative-ring.principal-ideal*. We abbreviate the principal ideal (p) of $F[X]$.

The principal ideal generated by p in $F[X]$, written out concretely.

definition *poly-pideal* :: $(nat \Rightarrow 'a) \Rightarrow (nat \Rightarrow 'a) \text{ set}$

where *poly-pideal* $p = \{r \otimes_P p \mid r. r \in \text{poly-carrier}\}$

lemma *poly-pideal-memI*: $r \in \text{poly-carrier} \implies r \otimes_P p \in \text{poly-pideal } p$

<proof>

lemma *poly-pideal-contains*:

assumes $p \in \text{poly-carrier}$ **shows** $p \in \text{poly-pideal } p$

<proof>

poly-pideal agrees with the generic *commutative-ring.principal-ideal* of the polynomial ring, and is therefore an ideal of $F[X]$.

lemma *poly-pideal-is-ideal*:

assumes $p \in \text{poly-carrier}$

shows *Ideal* (*poly-pideal* p) *poly-carrier* $(\oplus_P) (\otimes_P) \mathbf{0}_P \mathbf{1}_P$

<proof>

Kronecker's lemma. For an irreducible p , the principal ideal (p) is maximal in $F[X]$. Properness holds because $\mathbf{1}_P$ would force p to be a unit; maximality uses that $F[X]$ is a PID (*poly-PID*): any ideal J between (p) and $F[X]$ is principal, $J = \text{poly-pideal } b$, and $p = c \otimes_P b$ forces (by irreducibility) either b a unit, so $J = \text{poly-carrier}$, or c a unit, so $J \subseteq \text{poly-pideal } p$.

theorem *poly-irreducible-maximal*:

assumes *irr*: *poly-irreducible p*
shows *Ideal.maximal-ideal (poly-pideal p) poly-carrier ($\oplus_P$) ($\otimes_P$) $\mathbf{0}_P$ $\mathbf{1}_P$*
<proof>

The principal ideal (p) is an ideal of the *commutative* polynomial ring, so the quotient theory of *ideal-in-comm-ring* applies to it.

lemma *poly-pideal-in-comm-ring*:
assumes *p*: *p* $\in$ *poly-carrier*
shows *ideal-in-comm-ring (poly-pideal p) poly-carrier ($\oplus_P$) ($\otimes_P$) $\mathbf{0}_P$ $\mathbf{1}_P$*
<proof>

Kronecker's theorem. For an irreducible p , the principal ideal (p) is a maximal ideal of the *commutative* polynomial ring $F[X]$. By the maximal-ideal characterisation *ideal-in-comm-ring.maximal-ideal-iff-quotient-field*, this is exactly the statement that the quotient $F[X]/(p)$ is a field: it is nontrivial and every nonzero coset is invertible.

theorem *poly-quotient-field*:
assumes *irr*: *poly-irreducible p*
shows *ideal-in-comm-ring (poly-pideal p) poly-carrier ($\oplus_P$) ($\otimes_P$) $\mathbf{0}_P$ $\mathbf{1}_P$*
 $\wedge$ *Ideal.maximal-ideal (poly-pideal p) poly-carrier ($\oplus_P$) ($\otimes_P$) $\mathbf{0}_P$ $\mathbf{1}_P$*
<proof>

50.1 The size of the quotient ring

Membership in the principal ideal (p) is detected by the remainder: x is a multiple of p iff $p \text{ mod } x = \mathbf{0}_P$.

lemma *poly-pideal-iff-pmod*:
assumes *p*: *p* $\in$ *poly-carrier* **and** *pnz*: *p* $\neq \mathbf{0}_P$ **and** *x*: *x* $\in$ *poly-carrier*
shows *x* $\in$ *poly-pideal p* $\longleftrightarrow$ *p mod x = 0_P*
<proof>

The coset map $r \mapsto \text{coset } r$ restricted to *low-poly (degree p)* is a bijection onto the set of cosets: every coset has a unique representative of degree below p . Injectivity is uniqueness of *pmod* (equal cosets have equal remainders, and a low-degree polynomial is its own remainder).

theorem *bij-betw-low-poly-cosets*:
assumes *p*: *p* $\in$ *poly-carrier* **and** *pnz*: *p* $\neq \mathbf{0}_P$
shows *bij-betw (Ideal.coset (poly-pideal p) poly-carrier ($\oplus_P$) $\mathbf{0}_P$)*
 $(\text{low-poly } (\text{degree } p))$
 $(\text{Ideal.quotient-set } (\text{poly-pideal } p) \text{ poly-carrier } (\oplus_P) \mathbf{0}_P)$
<proof>

The quotient $F[X]/(p)$ has $|F|^{\text{deg } p}$ elements when the field F is finite: the cosets are in bijection with the $(\text{card } R)^{\text{degree } p}$ polynomials of degree below p .

theorem *card-poly-quotient*:
assumes *finite R* *p* $\in$ *poly-carrier* *p* $\neq \mathbf{0}_P$

shows $\text{card } (\text{Ideal.quotient-set } (\text{poly-pideal } p) \text{ poly-carrier } (\oplus_P) \mathbf{0}_P) = \text{card } R \wedge$
degree p
 $\langle \text{proof} \rangle$

50.2 Divisibility and Euclid's lemma

Divisibility in $F[X]$: d divides x when x is a multiple of d . This is exactly membership of x in the principal ideal *poly-pideal* d .

definition *poly-dvd* :: $(\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow \text{bool}$
where *poly-dvd* $d \ x \longleftrightarrow (\exists q. q \in \text{poly-carrier} \wedge x = q \otimes_P d)$

lemma *poly-dvd-iff-mem-pideal*: *poly-dvd* $d \ x \longleftrightarrow x \in \text{poly-pideal } d$
 $\langle \text{proof} \rangle$

A concrete divisibility witness with a polynomial divisor necessarily has a polynomial dividend. This recovers the carrier condition after working with the ambient function type.

lemma *poly-dvd-closed*:
assumes *dvd*: *poly-dvd* $d \ x$ **and** $d: d \in \text{poly-carrier}$
shows $x \in \text{poly-carrier}$
 $\langle \text{proof} \rangle$

For a nonzero divisor, concrete polynomial divisibility is equivalent to having zero remainder under the polynomial division algorithm.

lemma *poly-dvd-iff-pmod-zero*:
assumes $d: d \in \text{poly-carrier}$ **and** *dnz*: $d \neq \mathbf{0}_P$
and $x: x \in \text{poly-carrier}$
shows *poly-dvd* $d \ x \longleftrightarrow \text{pmod } x \ d = \mathbf{0}_P$
 $\langle \text{proof} \rangle$

Euclid's lemma / irreducibles are prime. For an irreducible p , if p divides a product $a \otimes_P b$ then it divides one of the factors. This is primality of the maximal ideal *poly-pideal* p (*ideal-in-comm-ring.maximal-imp-prime-ideal* applied to *poly-irreducible-maximal*).

theorem *poly-irreducible-prime*:
assumes *irr*: *poly-irreducible* p
and $a: a \in \text{poly-carrier}$ **and** $b: b \in \text{poly-carrier}$
and *dvd*: *poly-dvd* $p \ (a \otimes_P b)$
shows *poly-dvd* $p \ a \vee \text{poly-dvd } p \ b$
 $\langle \text{proof} \rangle$

An irreducible polynomial does not divide $\mathbf{1}_P$ (else it would be a unit).

lemma *poly-irreducible-not-dvd-one*:
assumes *irr*: *poly-irreducible* p **shows** $\neg \text{poly-dvd } p \ \mathbf{1}_P$
 $\langle \text{proof} \rangle$

Euclid's lemma for products. If an irreducible p divides a product of a list of polynomials, it divides one of them. (Base case: it cannot divide the empty product $\mathbf{1}_P$; step: split off the head with *poly-irreducible-prime*.)

lemma *poly-irreducible-prime-list*:
assumes *irr*: *poly-irreducible p*
shows $\llbracket \forall f \in \text{set } fs. f \in \text{poly-carrier}; \text{poly-dvd } p (\text{poly-prod } fs) \rrbracket$
 $\implies \exists f \in \text{set } fs. \text{poly-dvd } p f$
 $\langle \text{proof} \rangle$

50.3 Uniqueness of the irreducible factorisation

A nonzero divisor has degree at most that of its multiple.

lemma *poly-dvd-degree-le*:
assumes *dvd*: *poly-dvd d x* **and** *dc*: *d* $\in$ *poly-carrier* **and** *xnz*: *x* $\neq 0_P$
shows *degree d* $\leq$ *degree x*
 $\langle \text{proof} \rangle$

Two monic irreducibles that divide one another are equal: *p* dividing the irreducible *q* makes *p* a unit or an associate of *q*; being a non-unit monic, it equals the monic *q*.

lemma *monic-irreducible-dvd-eq*:
assumes *ip*: *poly-irreducible p* **and** *iq*: *poly-irreducible q*
and *mp*: *poly-monic p* **and** *mq*: *poly-monic q* **and** *dvd*: *poly-dvd p q*
shows *p* = *q*
 $\langle \text{proof} \rangle$

Pulling a factor out of a product: if *g* occurs in *gs*, the product factors as *g* times the product of the rest.

lemma *poly-prod-remove1*:
assumes *g* $\in$ *set gs* **and** $\forall f \in \text{set } gs. f \in \text{poly-carrier}$
shows *poly-prod gs* = *g* $\otimes_P$ *poly-prod (remove1 g gs)*
 $\langle \text{proof} \rangle$

Uniqueness of the monic irreducible factorisation. Two lists of monic irreducibles with equal products are permutations of one another. By induction on the first list: the head *f* divides the other product, hence equals some monic irreducible *g* in it; cancel *f* and recurse on the remaining factors.

theorem *factorization-unique*:
 $\llbracket \forall f \in \text{set } fs. \text{poly-irreducible } f \wedge \text{poly-monic } f;$
 $\forall g \in \text{set } gs. \text{poly-irreducible } g \wedge \text{poly-monic } g;$
 $\text{poly-prod } fs = \text{poly-prod } gs \rrbracket \implies \text{mset } fs = \text{mset } gs$
 $\langle \text{proof} \rangle$

end

end

51 The prime fields $GF(p) = \mathbb{Z}/p\mathbb{Z}$

theory *GF-p*

```

imports Ring-Theory Poly-Ring Poly-Ideal HOL-Computational-Algebra.Primes
begin

```

51.1 The residue rings $\mathbb{Z}/n\mathbb{Z}$

Ring-Theory suppresses HOL's arithmetic notation; restore it locally for the modular arithmetic below.

```

notation plus (infixl <+> 65)
notation minus (infixl <-> 65)
unbundle uminus-syntax

```

For any $1 < n$, the integers modulo n form a commutative ring: carrier $\{0..<n\}$ of *nat*, arithmetic modulo n . This is the abstract residue ring $\mathbb{Z}/n\mathbb{Z}$; when n is prime it is in fact a field, developed in *GF-p*. The bound $1 < n$ ensures both 0 and 1 lie in $\{0..<n\}$ and are distinct in *nat*; the trivial ring case $n = 1$ is not modelled by this carrier.

```

definition gfp-add :: nat ⇒ nat ⇒ nat ⇒ nat
  where gfp-add n x y = (x + y) mod n

```

```

definition gfp-mult :: nat ⇒ nat ⇒ nat ⇒ nat
  where gfp-mult n x y = (x * y) mod n

```

```

context
  fixes n :: nat
  assumes n-gt-1: n > 1
begin

```

```

lemma n-gt-0: n > 0 <proof>

```

Closure: results of modular arithmetic land in $\{0..<n\}$.

```

lemma gfp-add-closed [simp]: gfp-add n x y ∈ {0..<n}
  <proof>

```

```

lemma gfp-mult-closed [simp]: gfp-mult n x y ∈ {0..<n}
  <proof>

```

```

lemma zero-mem: (0::nat) ∈ {0..<n} and one-mem: (1::nat) ∈ {0..<n}
  <proof>

```

The additive part is an abelian group: associative and commutative modular addition, with $n - x$ (reduced) as the inverse of x .

```

lemma gfp-add-group: Group {0..<n} (gfp-add n) 0
  <proof>

```

```

lemma gfp-add-commutative-monoid: commutative-monoid {0..<n} (gfp-add n) 0
  <proof>

```

```

lemma gfp-additive: Abelian-Group {0..<n} (gfp-add n) 0

```

⟨proof⟩

The multiplicative part is a commutative monoid with unit 1 .

lemma *gfp-multiplicative: commutative-monoid* $\{0..<n\}$ (*gfp-mult* n) 1
 ⟨proof⟩

lemma *zfact-ring: Ring* $\{0..<n\}$ (*gfp-add* n) (*gfp-mult* n) 0 1
 ⟨proof⟩

lemma *zfact-commutative-ring: commutative-ring* $\{0..<n\}$ (*gfp-add* n) (*gfp-mult* n)
 0 1
 ⟨proof⟩

The residue ring $\mathbb{Z}/n\mathbb{Z}$ has exactly n elements.

lemma *card-zfact: card* $\{0..<n\} = n$ ⟨proof⟩

end

Hence, for every $1 < n$, the locale *commutative-ring* is non-vacuous with a witness of cardinality n .

51.2 The residue ring as a field

For a prime p , the residue ring *ZFact* becomes a field: the only extra step over the general $0 < n$ case is that every nonzero residue has a multiplicative inverse, which follows from Bézout's identity since p is prime.

Carrier $\{0..<p\}$ of *nat*, arithmetic modulo p , arithmetic operations *gfp-add* and *gfp-mult* shared with the ring-level development in *ZFact*.

context

fixes $p :: nat$

assumes *prime-p: prime* p

begin

lemma *p-gt-1: $p > 1$* ⟨proof⟩

lemma *p-gt-0: $p > 0$* ⟨proof⟩

Inverses. For $a \in \{1..<p\}$, a is coprime to the prime p , so Bézout gives integers with $u \cdot a + v \cdot p = 1$; reducing u modulo p yields a multiplicative inverse of a .

lemma *gfp-inverse-exists:*

assumes $a: a \in \{0..<p\}$ **and** $anz: a \neq 0$

shows $\exists b \in \{0..<p\}. \text{gfp-mult } p \ a \ b = 1$

⟨proof⟩

lemma *gfp-field: Field* $\{0..<p\}$ (*gfp-add* p) (*gfp-mult* p) 0 1
 ⟨proof⟩

The prime field has exactly p elements.

lemma *card-gfp*: $\text{card } \{0..<p\} = p$ *<proof>*

end

Hence prime fields of every prime order exist — in particular the *Field* locale is non-vacuous for every prime cardinality.

51.3 The two-element field as a non-vacuity witness

A concrete field with two elements, witnessing that the *Field* and *integral-domain* locales (and hence the abstract *'a ring* type as a field) are non-vacuous. Carrier $\{0, 1\}$ of *nat*.

definition *gf2-add* :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat}$ **where** *gf2-add* $x\ y \equiv (x + y) \bmod 2$

definition *gf2-mult* :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat}$ **where** *gf2-mult* $x\ y \equiv (x * y) \bmod 2$

Enumeration helper: membership in the two-element carrier.

lemma *gf2-cases*: $x \in \{0, 1::\text{nat}\} \implies x = 0 \vee x = 1$ *<proof>*

lemma *gf2-add-simps* [*simp*]:
 $\text{gf2-add } 0\ 0 = 0$ $\text{gf2-add } 0\ 1 = 1$ $\text{gf2-add } 1\ 0 = 1$ $\text{gf2-add } 1\ 1 = 0$
<proof>

lemma *gf2-mult-simps* [*simp*]:
 $\text{gf2-mult } 0\ 0 = 0$ $\text{gf2-mult } 0\ 1 = 0$ $\text{gf2-mult } 1\ 0 = 0$ $\text{gf2-mult } 1\ 1 = 1$
<proof>

lemma *gf2-add-closed* [*simp*]: $\llbracket x \in \{0, 1::\text{nat}\}; y \in \{0, 1\} \rrbracket \implies \text{gf2-add } x\ y \in \{0, 1\}$
<proof>

lemma *gf2-mult-closed* [*simp*]: $\llbracket x \in \{0, 1::\text{nat}\}; y \in \{0, 1\} \rrbracket \implies \text{gf2-mult } x\ y \in \{0, 1\}$
<proof>

The additive part is a group (each element is its own inverse), via *GroupI*.

lemma *gf2-add-group*: *Group* $\{0, 1::\text{nat}\}$ *gf2-add* 0
<proof>

lemma *gf2-add-commutative-monoid*: *commutative-monoid* $\{0, 1::\text{nat}\}$ *gf2-add* 0
<proof>

The additive part is therefore an abelian group.

lemma *gf2-additive*: *Abelian-Group* $\{0, 1::\text{nat}\}$ *gf2-add* 0
<proof>

The multiplicative part is a commutative monoid.

lemma *gf2-multiplicative*: *commutative-monoid* $\{0, 1::\text{nat}\}$ *gf2-mult* 1

$\langle \text{proof} \rangle$

lemma *gf2-ring*: *Ring* $\{0, 1::\text{nat}\}$ *gf2-add gf2-mult 0 1*
 $\langle \text{proof} \rangle$

lemma *gf2-commutative-ring*: *commutative-ring* $\{0, 1::\text{nat}\}$ *gf2-add gf2-mult 0 1*
 $\langle \text{proof} \rangle$

The two-element field. This is the single witness the downstream constructions consume (via *interpretation* ... by (rule *gf2-field*) in *GF4* and *GF8*); it also witnesses non-vacuity of the *Field*, *integral-domain* and *commutative-ring* locales, the last two by the locale hierarchy (a *Field* is in particular an *integral-domain*).

lemma *gf2-field*: *Field* $\{0, 1::\text{nat}\}$ *gf2-add gf2-mult 0 1*
 $\langle \text{proof} \rangle$

51.4 The two-element field with polynomial machinery

interpretation *GF2*: *Field* $\{0, 1::\text{nat}\}$ *gf2-add gf2-mult 0 1*
 $\langle \text{proof} \rangle$

lemma *GF2-poly-const-carrier* [*simp*, *intro*]: *GF2.poly-const 1* $\in$ *GF2.poly-carrier*
 $\langle \text{proof} \rangle$

lemma *GF2-monom-carrier* [*simp*, *intro*]: *GF2.monom 1 n* $\in$ *GF2.poly-carrier*
 $\langle \text{proof} \rangle$

51.5 The 3-element prime field with polynomial machinery

A single, canonical interpretation of the *Field* locale on *GF(3)*.

interpretation *GF3*: *Field* $\{0..<3::\text{nat}\}$ *gfp-add 3 gfp-mult 3 0 1*
 $\langle \text{proof} \rangle$

lemma *GF3-poly-const-carrier* [*simp*, *intro*]: *GF3.poly-const 1* $\in$ *GF3.poly-carrier*
 $\langle \text{proof} \rangle$

lemma *GF3-monom-carrier* [*simp*, *intro*]: *GF3.monom 1 n* $\in$ *GF3.poly-carrier*
 $\langle \text{proof} \rangle$

An end-to-end witness for the carrier-set field machinery, over a non-binary prime field: the polynomial $X^2 + 1$ is irreducible over *GF(3)* (it has no root there, as -1 is not a square modulo 3), so by Kronecker's theorem the quotient $\text{GF}(3)[X] / (X^2 + 1)$ is a field, and by *Field.card-poly-quotient* it has $3^2 = 9$ elements — the field *GF(9)*.

The witness polynomial $\Psi = X^2 + 1$ over *GF(3)*.

definition *Psi* :: *nat* $\Rightarrow$ *nat*
where *Psi* $\equiv$ *GF3.poly-add (GF3.monom 1 2) (GF3.poly-const 1)*

lemma *Psi-coeff*: $Psi\ i = (if\ i = 0 \vee i = 2\ then\ 1\ else\ 0)$
 $\langle proof \rangle$

lemma *Psi-closed*: $Psi \in GF3.poly-carrier$
 $\langle proof \rangle$

Psi has degree exactly 2.

lemma *degree-Psi*: $GF3.degree\ Psi = 2$
 $\langle proof \rangle$

Psi has no root in $\{0..<3\}$: $\alpha^2 + 1$ is 1, 2, 2 at $\alpha = 0, 1, 2$ respectively — never 0.

lemma *Psi-no-root*: $\alpha \in \{0..<3::nat\} \implies GF3.eval\ \alpha\ Psi \neq 0$
 $\langle proof \rangle$

Therefore *Psi* is irreducible over $GF(3)$.

theorem *Psi-irreducible*: $GF3.poly-irreducible\ Psi$
 $\langle proof \rangle$

51.6 The four-element field $GF(4)$ via Kronecker's construction

The polynomial $X^2 + X + 1$ is irreducible over the two-element field $GF(2)$ (it has no root there), so by Kronecker's theorem (*Field.poly-quotient-field*) the quotient $GF(2)[X] / (X^2 + X + 1)$ is a field — the four-element field $GF(4)$.

The witness polynomial $\Phi = X^2 + X + 1$ over $GF(2)$.

definition *Phi* :: $nat \Rightarrow nat$
where $Phi \equiv GF2.poly-add\ (GF2.poly-add\ (GF2.monom\ 1\ 2)\ (GF2.monom\ 1\ 1))\ (GF2.poly-const\ 1)$

Its coefficients: 1 at degrees 0, 1, 2, and 0 above.

lemma *Phi-coeff*: $Phi\ i = (if\ i \leq 2\ then\ 1\ else\ 0)$
 $\langle proof \rangle$

lemma *Phi-closed*: $Phi \in GF2.poly-carrier$
 $\langle proof \rangle$

Phi has degree exactly 2.

lemma *degree-Phi*: $GF2.degree\ Phi = 2$
 $\langle proof \rangle$

Phi has no root in $\{0, 1\}$: both candidate values evaluate to 1, since $X^2 + X + 1$ at 0 is 1 and at 1 is $1 + 1 + 1 = 1$ modulo 2.

lemma *Phi-no-root*:
assumes $a: \alpha \in \{0, 1::nat\}$

shows $GF2.eval \ \alpha \ \Phi \neq 0$
 $\langle proof \rangle$

Therefore Φ is irreducible over $GF(2)$.

theorem Φ -irreducible: $GF2.poly\text{-}irreducible \ \Phi$
 $\langle proof \rangle$

Kronecker. The principal ideal (Φ) is maximal in $GF(2)[X]$, so the quotient $GF(2)[X]/(\Phi)$ is a field — a four-element field, $GF(4)$.

theorem $GF4$ -is-field:
ideal-in-comm-ring ($GF2.poly\text{-}pideal \ \Phi$) $GF2.poly\text{-}carrier \ GF2.poly\text{-}add \ GF2.poly\text{-}mult$
 $GF2.poly\text{-}zero \ GF2.poly\text{-}one$
 $\wedge$ *Ideal.maximal-ideal* ($GF2.poly\text{-}pideal \ \Phi$) $GF2.poly\text{-}carrier \ GF2.poly\text{-}add \ GF2.poly\text{-}mult$
 $GF2.poly\text{-}zero \ GF2.poly\text{-}one$
 $\langle proof \rangle$

51.7 The eight-element field $GF(8)$ via Kronecker's construction

A cubic witness for the carrier-set field machinery: the polynomial $X^3 + X + 1$ is irreducible over $GF(2)$ (degree 3 with no root), so by Kronecker's theorem the quotient $GF(2)[X] / (X^3 + X + 1)$ is a field, and by *Field.card-poly-quotient* it has $2^3 = 8$ elements — the field $GF(8)$.

The witness polynomial $\Theta = X^3 + X + 1$ over $GF(2)$.

definition $\Theta :: nat \Rightarrow nat$
where $\Theta = GF2.poly\text{-}add \ (GF2.poly\text{-}add \ (GF2.monom \ 1 \ 3) \ (GF2.monom \ 1 \ 1)) \ (GF2.poly\text{-}const \ 1)$

lemma Θ -coeff: $\Theta \ i = (if \ i = 0 \vee i = 1 \vee i = 3 \text{ then } 1 \text{ else } 0)$
 $\langle proof \rangle$

lemma Θ -closed: $\Theta \in GF2.poly\text{-}carrier$
 $\langle proof \rangle$

Θ has degree exactly 3.

lemma degree- Θ : $GF2.degree \ \Theta = 3$
 $\langle proof \rangle$

Θ has no root in $\{0, 1\}$: $\alpha^3 + \alpha + 1$ is 1 at both 0 and 1 (modulo 2).

lemma Θ -no-root: $\alpha \in \{0, 1 :: nat\} \implies GF2.eval \ \alpha \ \Theta \neq 0$
 $\langle proof \rangle$

Therefore Θ is irreducible over $GF(2)$.

theorem Θ -irreducible: $GF2.poly\text{-}irreducible \ \Theta$
 $\langle proof \rangle$

Kronecker. The quotient $GF(2)[X]/(\Theta)$ is a field with $2^3 = 8$ elements.

theorem *GF8-is-field*:

ideal-in-comm-ring (*GF2.poly-pideal Theta*) *GF2.poly-carrier* (*GF2.poly-add*) (*GF2.poly-mult*)
GF2.poly-zero *GF2.poly-one*
 $\wedge$ *Ideal.maximal-ideal* (*GF2.poly-pideal Theta*) *GF2.poly-carrier* (*GF2.poly-add*)
(*GF2.poly-mult*) *GF2.poly-zero* *GF2.poly-one*
 $\langle$ *proof* $\rangle$

theorem *card-GF8*:

card (*Ideal.quotient-set* (*GF2.poly-pideal Theta*) *GF2.poly-carrier* (*GF2.poly-add*)
GF2.poly-zero) = 8
 $\langle$ *proof* $\rangle$

51.8 The nine-element field $GF(9)$ via Kronecker's construction

Kronecker. The quotient $GF(3)[X]/(\Psi)$ is a field with $3^2 = 9$ elements.

theorem *GF9-is-field*:

ideal-in-comm-ring (*GF3.poly-pideal Psi*) *GF3.poly-carrier* (*GF3.poly-add*) (*GF3.poly-mult*)
GF3.poly-zero *GF3.poly-one*
 $\wedge$ *Ideal.maximal-ideal* (*GF3.poly-pideal Psi*) *GF3.poly-carrier* (*GF3.poly-add*) (*GF3.poly-mult*)
GF3.poly-zero *GF3.poly-one*
 $\langle$ *proof* $\rangle$

theorem *card-GF9*:

card (*Ideal.quotient-set* (*GF3.poly-pideal Psi*) *GF3.poly-carrier* (*GF3.poly-add*) *GF3.poly-zero*)
= 9
 $\langle$ *proof* $\rangle$

end

52 Divisibility in commutative rings: units, associates, irreducibles, primes

theory *Divisibility-Theory*

imports *Ideal-Theory*

begin

Elementary multiplicative divisibility theory for a commutative ring, in the locale idiom of *Ring-Theory*. We work directly with ring elements rather than lifting the multiplicative monoid, which keeps the statements close to textbook divisibility.

context *commutative-ring*

begin

52.1 Divisibility and units

divides a b: *a* divides *b*, i.e. $b = a \cdot c$ for some ring element *c*.

definition *divides* :: 'a $\Rightarrow$ 'a $\Rightarrow$ bool
where *divides* a b $\equiv (\exists c \in R. b = a \cdot c)$

lemma *dividesI* [intro]: $\llbracket c \in R; b = a \cdot c \rrbracket \Longrightarrow \text{divides } a \ b$
 $\langle \text{proof} \rangle$

lemma *dividesE* [elim]:
assumes *divides* a b **obtains** c **where** $c \in R \ b = a \cdot c$
 $\langle \text{proof} \rangle$

lemma *divides-refl* [simp]: $a \in R \Longrightarrow \text{divides } a \ a$
 $\langle \text{proof} \rangle$

lemma *divides-trans* [trans]:
assumes *divides* a b *divides* b c **and** a: $a \in R$
shows *divides* a c
 $\langle \text{proof} \rangle$

lemma *divides-zero* [simp]: $a \in R \Longrightarrow \text{divides } a \ 0$
 $\langle \text{proof} \rangle$

lemma *one-divides* [simp]: $a \in R \Longrightarrow \text{divides } 1 \ a$
 $\langle \text{proof} \rangle$

lemma *divides-mult-right*:
assumes $b \in R$ **shows** *divides* a (a $\cdot$ b)
 $\langle \text{proof} \rangle$

A unit is an invertible element of the multiplicative monoid.

abbreviation *is-unit* :: 'a $\Rightarrow$ bool
where *is-unit* u $\equiv \text{multiplicative.invertible } u$

lemma *is-unitI* [intro]: $\llbracket u \cdot v = 1; v \cdot u = 1; v \in R \rrbracket \Longrightarrow \text{is-unit } u$
 $\langle \text{proof} \rangle$

lemma *unit-divides-one*:
assumes *is-unit* u **shows** *divides* u 1
 $\langle \text{proof} \rangle$

lemma *one-divides-unit-iff*:
assumes u: $u \in R$ **shows** *divides* u 1 $\longleftrightarrow \text{is-unit } u$
 $\langle \text{proof} \rangle$

lemma *unit-divides-all*:
assumes *is-unit* u $u \in R \ a \in R$ **shows** *divides* u a
 $\langle \text{proof} \rangle$

52.2 Associated elements

Two elements are associated if each divides the other.

definition *associated* :: 'a $\Rightarrow$ 'a $\Rightarrow$ bool
where *associated* a b $\longleftrightarrow$ divides a b $\wedge$ divides b a

lemma *associatedI* [intro]: $\llbracket \text{divides } a \ b; \text{divides } b \ a \rrbracket \Longrightarrow \text{associated } a \ b$
 $\langle \text{proof} \rangle$

lemma *associatedD1*: *associated* a b $\Longrightarrow$ divides a b
 $\langle \text{proof} \rangle$

lemma *associatedD2*: *associated* a b $\Longrightarrow$ divides b a
 $\langle \text{proof} \rangle$

lemma *associated-refl* [simp]: $a \in R \Longrightarrow \text{associated } a \ a$
 $\langle \text{proof} \rangle$

lemma *associated-sym*: *associated* a b $\Longrightarrow$ *associated* b a
 $\langle \text{proof} \rangle$

lemma *associated-trans* [trans]:
assumes *associated* a b *associated* b c $a \in R \ b \in R \ c \in R$
shows *associated* a c
 $\langle \text{proof} \rangle$

52.3 Greatest common divisors and least common multiples

is-gcd d a b: d is a common divisor of a and b that every common divisor divides. Gcds need not exist in a general ring, but when they do they are unique up to association; existence is established in a principal ideal domain (Bezout) below.

definition *is-gcd* :: 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ bool
where *is-gcd* d a b $\longleftrightarrow$
 $d \in R \wedge \text{divides } d \ a \wedge \text{divides } d \ b \wedge (\forall c \in R. \text{divides } c \ a \wedge \text{divides } c \ b \longrightarrow \text{divides } c \ d)$

is-lcm m a b: m is a common multiple of a and b dividing every common multiple.

definition *is-lcm* :: 'a $\Rightarrow$ 'a $\Rightarrow$ 'a $\Rightarrow$ bool
where *is-lcm* m a b $\longleftrightarrow$
 $m \in R \wedge \text{divides } a \ m \wedge \text{divides } b \ m \wedge (\forall c \in R. \text{divides } a \ c \wedge \text{divides } b \ c \longrightarrow \text{divides } m \ c)$

lemma *is-gcdI*:
 $\llbracket d \in R; \text{divides } d \ a; \text{divides } d \ b; \bigwedge c. \llbracket c \in R; \text{divides } c \ a; \text{divides } c \ b \rrbracket \Longrightarrow \text{divides } c \ d \rrbracket \Longrightarrow \text{is-gcd } d \ a \ b$
 $\langle \text{proof} \rangle$

lemma *is-gcdD*:
assumes *is-gcd d a b*
shows $d \in R$ divides d *a* divides d *b*
and $\bigwedge c. \llbracket c \in R; \text{divides } c \text{ } a; \text{divides } c \text{ } b \rrbracket \implies \text{divides } c \text{ } d$
 $\langle \text{proof} \rangle$

lemma *is-gcd-sym*: *is-gcd d a b* $\implies$ *is-gcd d b a*
 $\langle \text{proof} \rangle$

lemma *is-lcm-sym*: *is-lcm m a b* $\implies$ *is-lcm m b a*
 $\langle \text{proof} \rangle$

Any two gcDs of the same pair are associated.

lemma *is-gcd-unique*:
assumes *is-gcd d a b* **and** *is-gcd d' a b*
shows *associated d d'*
 $\langle \text{proof} \rangle$

Dually, any two lcMs are associated.

lemma *is-lcm-unique*:
assumes *is-lcm m a b* **and** *is-lcm m' a b*
shows *associated m m'*
 $\langle \text{proof} \rangle$

end

52.4 Irreducible and prime elements

context *commutative-ring*
begin

An irreducible element is a nonzero non-unit whose only factorisations are trivial.

definition *irreducible-elem* :: $'a \Rightarrow \text{bool}$
where *irreducible-elem* $p \longleftrightarrow$
 $p \in R \wedge p \neq \mathbf{0} \wedge \neg \text{is-unit } p \wedge$
 $(\forall a \in R. \forall b \in R. p = a \cdot b \longrightarrow \text{is-unit } a \vee \text{is-unit } b)$

lemma *irreducible-elemI*:
assumes $p \in R$ $p \neq \mathbf{0}$ $\neg \text{is-unit } p$
and $\bigwedge a \text{ } b. \llbracket a \in R; b \in R; p = a \cdot b \rrbracket \implies \text{is-unit } a \vee \text{is-unit } b$
shows *irreducible-elem* p
 $\langle \text{proof} \rangle$

lemma *irreducible-elemD*:
assumes *irreducible-elem* p
shows $p \in R$ $p \neq \mathbf{0}$ $\neg \text{is-unit } p$
and $\bigwedge a \text{ } b. \llbracket a \in R; b \in R; p = a \cdot b \rrbracket \implies \text{is-unit } a \vee \text{is-unit } b$
 $\langle \text{proof} \rangle$

A prime element is a nonzero non-unit that, when dividing a product, divides a factor.

definition *prime-elem* :: 'a $\Rightarrow$ bool
where *prime-elem* $p \longleftrightarrow$
 $p \in R \wedge p \neq \mathbf{0} \wedge \neg \text{is-unit } p \wedge$
 $(\forall a \in R. \forall b \in R. \text{divides } p (a \cdot b) \longrightarrow \text{divides } p a \vee \text{divides } p b)$

lemma *prime-elemI*:
assumes $p \in R$ $p \neq \mathbf{0}$ $\neg \text{is-unit } p$
and $\bigwedge a b. [\![a \in R; b \in R; \text{divides } p (a \cdot b)]\!] \Longrightarrow \text{divides } p a \vee \text{divides } p b$
shows *prime-elem* p
 $\langle \text{proof} \rangle$

lemma *prime-elemD*:
assumes *prime-elem* p
shows $p \in R$ $p \neq \mathbf{0}$ $\neg \text{is-unit } p$
and $\bigwedge a b. [\![a \in R; b \in R; \text{divides } p (a \cdot b)]\!] \Longrightarrow \text{divides } p a \vee \text{divides } p b$
 $\langle \text{proof} \rangle$

end

52.5 In an integral domain, prime elements are irreducible

context *integral-domain*
begin

Cancellation of a nonzero factor: the defining property that separates domains from general commutative rings. From $c \cdot a = c \cdot b$ with $c \neq \mathbf{0}$ we get $c \cdot (a - b) = \mathbf{0}$, and absence of zero divisors forces $a - b = \mathbf{0}$.

lemma *mult-cancel-left*:
assumes $c: c \in R$ **and** $a: a \in R$ **and** $b: b \in R$ **and** $\text{cnz}: c \neq \mathbf{0}$ **and** $\text{eq}: c \cdot a = c \cdot b$
shows $a = b$
 $\langle \text{proof} \rangle$

The classical characterization of association in a domain: a and b are associated iff they differ by a unit factor, $b = u \cdot a$. (The right-to-left direction holds in any commutative ring; the forward direction is where cancellation — hence the domain hypothesis — is used.)

lemma *associated-iff-unit-multiple*:
assumes $a: a \in R$ **and** $b: b \in R$
shows *associated* $a b \longleftrightarrow (\exists u. \text{is-unit } u \wedge u \in R \wedge b = u \cdot a)$
 $\langle \text{proof} \rangle$

lemma *prime-imp-irreducible*:
assumes $p: \text{prime-elem } p$
shows *irreducible-elem* p
 $\langle \text{proof} \rangle$

end

52.6 Divisibility via principal ideals

Divisibility is containment of principal ideals the other way round: a divides b iff $(b) \subseteq (a)$ iff $b \in (a)$. This is the bridge that turns the ideal-theoretic PID argument into statements about ring elements.

context *commutative-ring*
begin

lemma *divides-iff-mem-principal*:
assumes $a: a \in R$ **and** $b: b \in R$
shows $\text{divides } a \ b \longleftrightarrow b \in \text{principal-ideal } a$
 $\langle \text{proof} \rangle$

lemma *divides-iff-principal-subset*:
assumes $a: a \in R$ **and** $b: b \in R$
shows $\text{divides } a \ b \longleftrightarrow \text{principal-ideal } b \subseteq \text{principal-ideal } a$
 $\langle \text{proof} \rangle$

Associated elements generate the same principal ideal.

lemma *associated-iff-same-principal*:
assumes $a: a \in R$ **and** $b: b \in R$
shows $\text{associated } a \ b \longleftrightarrow \text{principal-ideal } a = \text{principal-ideal } b$
 $\langle \text{proof} \rangle$

An ideal containing $\mathbf{1}$ is the whole ring.

lemma *ideal-contains-one-eq-whole*:
assumes $I: \text{Ideal } I \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ **and** $\text{one}: \mathbf{1} \in I$
shows $I = R$
 $\langle \text{proof} \rangle$

A principal ideal is the whole ring exactly when its generator is a unit.

lemma *principal-ideal-eq-whole-iff-unit*:
assumes $a: a \in R$
shows $\text{principal-ideal } a = R \longleftrightarrow \text{is-unit } a$
 $\langle \text{proof} \rangle$

end

52.7 Noetherian rings

A (two-sided) noetherian ring is one in which every ascending chain of ideals eventually stabilises. This is the ascending-chain formulation of noetherianity, and is the form used by the factorisation arguments below.

locale *Noetherian-Ring* = *Ring* +

assumes *ideal-chain-stabilises*:

$\llbracket \bigwedge n. \text{Ideal } (A\ n) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}; \bigwedge n. A\ n \subseteq A\ (\text{Suc } n) \rrbracket \implies$
 $\exists N. \forall n. N \leq n \longrightarrow A\ n = A\ N$

A noetherian domain is both noetherian and an integral domain.

locale *Noetherian-Domain* = *Noetherian-Ring* + *integral-domain*

52.8 Principal ideal domains

A principal ideal domain (PID) is an integral domain in which every ideal is principal.

locale *principal-ideal-domain* = *integral-domain* +

assumes *principal*: *Ideal* *I* *R* $(+)$ $(\cdot)$ $\mathbf{0} \ \mathbf{1} \implies \exists a \in R. I = \text{principal-ideal } a$

context *principal-ideal-domain*

begin

In a PID an irreducible element generates a maximal ideal. We use the maximality criterion of *Ideal*: any ideal J containing (p) is principal, $J = (q)$, so q divides p ; by irreducibility one factor is a unit, giving $J = (p)$ or $J = R$.

theorem *irreducible-imp-maximal-ideal*:

assumes *p*: *irreducible-elem* *p*

shows *Ideal.maximal-ideal* (*principal-ideal* *p*) *R* $(+)$ $(\cdot)$ $\mathbf{0} \ \mathbf{1}$

<proof>

Hence in a PID every irreducible element is prime (the converse of $\llbracket \text{integral-domain } ?R \ ?\text{addition} \ ?\text{multiplication} \ ?\text{zero} \ ?\text{unit}; \text{commutative-ring.prime-elem } ?R \ ?\text{multiplication} \ ?\text{zero} \ ?\text{unit} \ ?p \rrbracket \implies \text{commutative-ring.irreducible-elem } ?R \ ?\text{multiplication} \ ?\text{zero} \ ?\text{unit} \ ?p$).

theorem *irreducible-imp-prime*:

assumes *p*: *irreducible-elem* *p*

shows *prime-elem* *p*

<proof>

52.8.1 The ascending chain condition on principal ideals

The union of a countable ascending chain of ideals is an ideal.

lemma *Union-chain-ideal*:

assumes *I*: $\bigwedge n. \text{Ideal } (A\ n) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

and *mono*: $\bigwedge n. A\ n \subseteq A\ (\text{Suc } n)$

shows *Ideal* $(\bigcup n. A\ n) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

<proof>

Ascending chain condition: every ascending chain of (principal) ideals in a PID stabilises. Proof: the union is an ideal, hence principal = (c) ; the generator c lies in some $A\ N$, and from there on the chain cannot grow beyond $A\ N$.

theorem *ascending-chain-stabilises*:

assumes $I: \bigwedge n. \text{Ideal } (A\ n) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
and mono: $\bigwedge n. A\ n \subseteq A \ (Suc\ n)$
obtains N **where** $\bigwedge n. N \leq n \implies A\ n = A\ N$
 $\langle proof \rangle$

52.8.2 Bezout: existence of greatest common divisors

The ideal generated by two elements, $(a, b) = \{x \cdot a + y \cdot b\}$.

definition *bezout-ideal* :: $'a \Rightarrow 'a \Rightarrow 'a \text{ set}$
where *bezout-ideal* $a\ b \equiv \{x \cdot a + y \cdot b \mid x\ y. x \in R \wedge y \in R\}$

lemma *bezout-ideal-memI*:
 $\llbracket x \in R; y \in R \rrbracket \implies x \cdot a + y \cdot b \in \text{bezout-ideal } a\ b$
 $\langle proof \rangle$

lemma *bezout-ideal-is-ideal*:
assumes $a: a \in R$ **and** $b: b \in R$
shows *Ideal* (*bezout-ideal* $a\ b$) $R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
 $\langle proof \rangle$

a and b both lie in (a, b) .

lemma *a-in-bezout-ideal*: $\llbracket a \in R; b \in R \rrbracket \implies a \in \text{bezout-ideal } a\ b$
 $\langle proof \rangle$

lemma *b-in-bezout-ideal*: $\llbracket a \in R; b \in R \rrbracket \implies b \in \text{bezout-ideal } a\ b$
 $\langle proof \rangle$

Bezout's theorem. The gcd generates the ideal (a, b) : divisibility of a and b is membership, and any common divisor divides every element of (a, b) , in particular d .

theorem *bezout*:
assumes $a: a \in R$ **and** $b: b \in R$
obtains $d\ x\ y$ **where** *is-gcd* $d\ a\ b$ $x \in R\ y \in R\ d = x \cdot a + y \cdot b$
 $\langle proof \rangle$

Consequently a greatest common divisor exists for every pair.

corollary *gcd-exists*:
assumes $a \in R\ b \in R$ **shows** $\exists d. \text{is-gcd } d\ a\ b$
 $\langle proof \rangle$

end

Principal ideal domains satisfy the noetherian ascending-chain condition.

sublocale *principal-ideal-domain* $\subseteq$ *Noetherian-Domain*
 $\langle proof \rangle$

52.9 Euclidean domains

A Euclidean domain is an integral domain equipped with a degree function φ into the naturals admitting division with remainder: for a, b nonzero there are q, r with $a = b \cdot q + r$ and either $r = \mathbf{0}$ or $\varphi r < \varphi b$.

```

locale Euclidean-Domain = integral-domain +
  fixes  $\varphi :: 'a \Rightarrow \text{nat}$ 
  assumes euclidean-division:
     $\llbracket a \in R; b \in R; a \neq \mathbf{0}; b \neq \mathbf{0} \rrbracket \Longrightarrow$ 
     $\exists q \in R. \exists r \in R. a = b \cdot q + r \wedge (r = \mathbf{0} \vee \varphi r < \varphi b)$ 
begin

```

Every Euclidean domain is a principal ideal domain: an ideal is generated by any nonzero element of least φ -value, since division with remainder against it lands back in the ideal and minimality forces a zero remainder.

```

sublocale principal-ideal-domain
  <proof>

```

end

A field is a Euclidean domain with the constant degree function (division with remainder is exact: $a = b \cdot (b^{-1} \cdot a) + \mathbf{0}$).

```

sublocale Field  $\subseteq$  Euclidean-Domain  $R (+) (\cdot) \mathbf{0} \mathbf{1} \lambda\cdot. 0$ 
  <proof>

```

52.10 Factorizations into irreducibles

The product of a list of ring elements.

```

context commutative-ring
begin

```

```

definition list-prod :: ' $a$  list  $\Rightarrow$  ' $a$ 
  where list-prod xs = foldr ( $\cdot$ ) xs  $\mathbf{1}$ 

```

```

lemma list-prod-Nil [simp]: list-prod [] =  $\mathbf{1}$ 
  <proof>

```

```

lemma list-prod-Cons [simp]: list-prod (x # xs) = x  $\cdot$  list-prod xs
  <proof>

```

```

lemma list-prod-closed [intro, simp]:
  set xs  $\subseteq R \Longrightarrow$  list-prod xs  $\in R$ 
  <proof>

```

A factorization of a is a list of irreducible ring elements whose product is a .

```

definition factorization :: ' $a$  list  $\Rightarrow$  ' $a \Rightarrow \text{bool}$ 
  where factorization fs a  $\equiv (\forall p \in \text{set fs. irreducible-elem } p) \wedge \text{list-prod fs} = a$ 

```

lemma *factorizationI*:

$\llbracket \bigwedge p. p \in \text{set } fs \implies \text{irreducible-elem } p; \text{list-prod } fs = a \rrbracket \implies \text{factorization } fs \ a$
 $\langle \text{proof} \rangle$

lemma *factorization-irreducible*: $\llbracket \text{factorization } fs \ a; p \in \text{set } fs \rrbracket \implies \text{irreducible-elem } p$
 $\langle \text{proof} \rangle$

lemma *factorization-prod*: $\text{factorization } fs \ a \implies \text{list-prod } fs = a$
 $\langle \text{proof} \rangle$

lemma *factorization-set-R*: $\text{factorization } fs \ a \implies \text{set } fs \subseteq R$
 $\langle \text{proof} \rangle$

lemma *list-prod-append* [*simp*]:
assumes $\text{set } xs \subseteq R \ \text{set } ys \subseteq R$
shows $\text{list-prod } (xs \ @ \ ys) = \text{list-prod } xs \cdot \text{list-prod } ys$
 $\langle \text{proof} \rangle$

An irreducible element has the one-item factorization $[p]$.

lemma *factorization-single*: $\text{irreducible-elem } p \implies \text{factorization } [p] \ p$
 $\langle \text{proof} \rangle$

Concatenating factorizations multiplies the factored elements.

lemma *factorization-append*:
assumes $\text{factorization } fs \ a \ \text{factorization } gs \ b$
shows $\text{factorization } (fs \ @ \ gs) \ (a \cdot b)$
 $\langle \text{proof} \rangle$

end

52.11 Uniqueness of factorization

context *integral-domain*
begin

A prime element dividing a list-product divides one of the factors.

lemma *prime-divides-list-prod*:
assumes $p: \text{prime-elem } p$ **and** $fs: \text{set } fs \subseteq R$ **and** $dvd: \text{divides } p \ (\text{list-prod } fs)$
shows $\exists f \in \text{set } fs. \text{divides } p \ f$
 $\langle \text{proof} \rangle$

If an irreducible q divides an irreducible p then they are associated.

lemma *irreducible-dvd-irreducible-assoc*:
assumes $q: \text{irreducible-elem } q$ **and** $p: \text{irreducible-elem } p$ **and** $dvd: \text{divides } q \ p$
shows $\text{associated } q \ p$
 $\langle \text{proof} \rangle$

Cancellation for association: a common nonzero factor may be dropped.

lemma *associated-cancel*:
assumes $c: c \in R$ **and** $x: x \in R$ **and** $y: y \in R$ **and** $cnz: c \neq 0$
and $as: \text{associated } (c \cdot x) (c \cdot y)$
shows $\text{associated } x y$
 $\langle \text{proof} \rangle$

Multiplying by a unit yields an associate.

lemma *associated-unit-mult*:
assumes $u: \text{is-unit } u$ $u \in R$ **and** $b: b \in R$
shows $\text{associated } (u \cdot b) b$
 $\langle \text{proof} \rangle$

Product of a list is unchanged by removing then re-multiplying a member (commutativity).

lemma *list-prod-remove1*:
assumes $\text{set } gs \subseteq R$ **and** $g \in \text{set } gs$
shows $\text{list-prod } gs = g \cdot \text{list-prod } (\text{remove1 } g \ gs)$
 $\langle \text{proof} \rangle$

Uniqueness of factorization (length form): any two prime factorizations of associated products have the same number of factors. This is the well-definedness of the number of irreducible factors — the core uniqueness statement of unique factorization.

theorem *prime-factorization-length-unique*:
assumes $\bigwedge f. f \in \text{set } fs \implies \text{prime-elem } f$ **and** $\bigwedge g. g \in \text{set } gs \implies \text{prime-elem } g$
and $\text{associated } (\text{list-prod } fs) (\text{list-prod } gs)$
shows $\text{length } fs = \text{length } gs$
 $\langle \text{proof} \rangle$

end

52.12 Factorial domains (unique factorization domains)

context *principal-ideal-domain*
begin

52.12.1 Existence of factorizations

Every nonzero non-unit of a PID admits a factorization into irreducibles. If not, the set of “bad” elements (nonzero non-units without a factorization) is nonempty; a bad element is not irreducible, so it factors into two non-units, at least one of which is bad and is a *proper* divisor. Iterating (dependent choice) yields a strictly ascending chain of principal ideals, contradicting the ascending chain condition.

theorem *factorization-exists*:
assumes $a: a \in R$ **and** $anz: a \neq 0$ **and** $anu: \neg \text{is-unit } a$

shows $\exists fs. \text{factorization } fs \ a$
 <proof>

end

A factorial domain (UFD) is an integral domain in which every nonzero non-unit factors into irreducibles, uniquely up to the number of factors (and, via $\llbracket \text{integral-domain } ?R \ ?\text{addition } ?\text{multiplication } ?\text{zero } ?\text{unit}; \bigwedge f. f \in \text{set } ?fs \implies \text{commutative-ring.prime-elem } ?R \ ?\text{multiplication } ?\text{zero } ?\text{unit } f; \bigwedge g. g \in \text{set } ?gs \implies \text{commutative-ring.prime-elem } ?R \ ?\text{multiplication } ?\text{zero } ?\text{unit } g; \text{commutative-ring.associated } ?R \ ?\text{multiplication } (\text{commutative-ring.list-prod } ?\text{multiplication } ?\text{unit } ?fs) (\text{commutative-ring.list-prod } ?\text{multiplication } ?\text{unit } ?gs) \rrbracket \implies \text{length } ?fs = \text{length } ?gs$, up to associates and order once the factors are prime).

locale *Factorial-Domain* = *integral-domain* +
assumes *factors-exist*: $\llbracket a \in R; a \neq 0; \neg \text{is-unit } a \rrbracket \implies \exists fs. \text{factorization } fs \ a$

Every principal ideal domain is a factorial domain: existence of factorizations is $\llbracket \text{principal-ideal-domain } ?R \ ?\text{addition } ?\text{multiplication } ?\text{zero } ?\text{unit}; ?a \in ?R; ?a \neq ?\text{zero}; \neg \text{Monoid.invertible } ?R \ ?\text{multiplication } ?\text{unit } ?a \rrbracket \implies \exists fs. \text{commutative-ring.factorization } ?R \ ?\text{multiplication } ?\text{zero } ?\text{unit } fs \ a$, established via the ascending chain condition.

sublocale *principal-ideal-domain* $\subseteq$ *Factorial-Domain*
 <proof>

52.13 Euclidean type classes as locale Euclidean domains; the integers

Isabelle's type class *euclidean-ring* (an *idom* with a *euclidean-size* and division with remainder) is exactly the type-class analogue of our locale *Euclidean-Domain*. The ring of integers $\mathbb{Z}$ is the primary instance.

Restore HOL arithmetic notation locally (*Ring-Theory* suppresses it).

notation *plus* (**infixl** $\langle + \rangle$ 65)
notation *minus* (**infixl** $\langle - \rangle$ 65)
unbundle *uminus-syntax*

52.14 The general type-class bridge

The division-with-remainder axiom of *Euclidean-Domain* is witnessed by $q = a \text{ div } b$, $r = a \text{ mod } b$: the identity $?b * (?a \text{ div } ?b) + ?a \text{ mod } ?b = ?a$ gives $a = b \cdot q + r$, and $?b \neq 0 \implies \text{euclidean-size } (?a \text{ mod } ?b) < \text{euclidean-size } ?b$ bounds the remainder's size. (The class's extra axioms — *euclidean-size* $0 = 0$, $?b \neq 0 \implies \text{euclidean-size } ?a \leq \text{euclidean-size } (?a * ?b)$ — exceed what the locale requires, so this is a one-directional bridge, which is all we need.)

lemma *euclidean-domain-TC*:
Euclidean-Domain (*UNIV* :: '*a* :: *euclidean-ring set*) (+) (*) 0 1 *euclidean-size*
 <proof>

52.15 The integers

$\mathbb{Z}$ is a Euclidean ring with Euclidean size the absolute value, so the bridge specialises to it directly. Through the interpretation $\mathbb{Z}$ inherits the whole tower: it is a principal ideal domain and a factorial (unique factorization) domain — the fundamental theorem of arithmetic.

interpretation *Int: Euclidean-Domain*

UNIV :: int set (+) () 0 1 euclidean-size :: int $\Rightarrow$ nat*
<proof>

Consequences, transported from the abstract tower.

lemmas *int-factorization-exists = Int.factorization-exists*

lemmas *int-irreducible-imp-prime = Int.irreducible-imp-prime*

lemmas *int-bezout = Int.bezout*

end

53 The prime subfield of a finite integral domain

theory *Prime-Subfield*

imports *Characteristic GF-p Divisibility-Theory*

begin

53.1 Finite integral domains

A finite integral domain is a field. Multiplication by a nonzero element is an injective self-map of the finite carrier, hence surjective; a preimage of the unit is a multiplicative inverse.

theorem (*in integral-domain*) *finite-integral-domain-imp-field:*

assumes *fn: finite R*

shows *Field R (+) ($\cdot$) 0 1*

<proof>

The characteristic map factors through the existing concrete field $GF(\text{characteristic})$. Its image is the canonical prime subfield of the finite domain. This is the bridge between the carrier-set ring hierarchy and the finite-field constructions.

locale *finite-integral-domain = integral-domain +*

assumes *finite-carrier: finite R*

begin

definition *prime-subfield :: 'a set*

where *prime-subfield = natmult ' {0.. characteristic }*

definition *characteristic-embedding :: nat $\Rightarrow$ 'a*

where *characteristic-embedding = restrict natmult {0.. characteristic }*

lemma *natmult-characteristic-multiple:*

shows $\text{natmult } (k * \text{characteristic}) = 0$
 $\langle \text{proof} \rangle$

lemma *natmult-mod-characteristic*:
shows $\text{natmult } (n \bmod \text{characteristic}) = \text{natmult } n$
 $\langle \text{proof} \rangle$

lemma *natmult-inj-on-characteristic*:
shows $\text{inj-on natmult } \{0..<\text{characteristic}\}$
 $\langle \text{proof} \rangle$

lemma *characteristic-embedding-add*:
assumes $m: m \in \{0..<\text{characteristic}\}$ **and** $n: n \in \{0..<\text{characteristic}\}$
shows $\text{characteristic-embedding } (\text{gfp-add } \text{characteristic } m \ n) =$
 $\text{characteristic-embedding } m + \text{characteristic-embedding } n$
 $\langle \text{proof} \rangle$

lemma *characteristic-embedding-mult*:
assumes $m: m \in \{0..<\text{characteristic}\}$ **and** $n: n \in \{0..<\text{characteristic}\}$
shows $\text{characteristic-embedding } (\text{gfp-mult } \text{characteristic } m \ n) =$
 $\text{characteristic-embedding } m \cdot \text{characteristic-embedding } n$
 $\langle \text{proof} \rangle$

theorem *characteristic-embedding-monomorphism*:
shows $\text{ring-monomorphism } \text{characteristic-embedding}$
 $\{0..<\text{characteristic}\} (\text{gfp-add } \text{characteristic}) (\text{gfp-mult } \text{characteristic}) \ 0 \ 1$
 $R \ (+) \ (\cdot) \ 0 \ 1$
 $\langle \text{proof} \rangle$

lemma *characteristic-embedding-image*:
 $\text{characteristic-embedding } \{0..<\text{characteristic}\} = \text{prime-subfield}$
 $\langle \text{proof} \rangle$

theorem *prime-subfield-subring*:
shows $\text{Subring } \text{prime-subfield } R \ (+) \ (\cdot) \ 0 \ 1$
 $\langle \text{proof} \rangle$

theorem *prime-subfield-field*:
shows $\text{Field } \text{prime-subfield } (+) \ (\cdot) \ 0 \ 1$
 $\langle \text{proof} \rangle$

theorem *card-prime-subfield*:
shows $\text{card } \text{prime-subfield} = \text{characteristic}$
 $\langle \text{proof} \rangle$

end

end

54 Modules over a ring

```

theory Module-Basics
  imports Ring-Theory Finite-Composite
begin

```

A (left) *module* over the ring R : an abelian group of elements M under $\oplus$ with zero $\mathbf{0}_M$, together with a scalar multiplication $\odot$ by ring elements satisfying the module axioms. This is the exact generalisation of theory *Vector-Space* from a field of scalars to a ring; a vector space is a module whose scalar ring is a field. We develop the elementary arithmetic (scaling by zero, by the ring negation, additivity of finite sums) and the notions of submodule and of a linear combination.

```

locale Module = Ring +
  fixes madd :: 'b  $\Rightarrow$  'b  $\Rightarrow$  'b (infixl  $\langle \oplus \rangle$  65)
    and mzero :: 'b ( $\langle \mathbf{0}_M \rangle$ )
    and mcarrier :: 'b set ( $\langle M \rangle$ )
    and scale :: 'a  $\Rightarrow$  'b  $\Rightarrow$  'b (infixr  $\langle \odot \rangle$  75)
  assumes madd-group: Abelian-Group  $M$  ( $\oplus$ )  $\mathbf{0}_M$ 
    and scale-closed:  $\llbracket a \in R; v \in M \rrbracket \Longrightarrow a \odot v \in M$ 
    and scale-distrib-madd:  $\llbracket a \in R; u \in M; v \in M \rrbracket \Longrightarrow a \odot (u \oplus v) = (a \odot u) \oplus (a \odot v)$ 
    and scale-distrib-add:  $\llbracket a \in R; b \in R; v \in M \rrbracket \Longrightarrow (a + b) \odot v = (a \odot v) \oplus (b \odot v)$ 
    and scale-scale:  $\llbracket a \in R; b \in R; v \in M \rrbracket \Longrightarrow (a \cdot b) \odot v = a \odot (b \odot v)$ 
    and scale-one:  $v \in M \Longrightarrow \mathbf{1} \odot v = v$ 
begin

```

The additive group of the module, available through the *madd* prefix.

```

sublocale madd: Abelian-Group  $M$  ( $\oplus$ )  $\mathbf{0}_M$ 
   $\langle \text{proof} \rangle$ 

```

54.1 Elementary closure and arithmetic

```

lemma mzero-closed [simp, intro]:  $\mathbf{0}_M \in M$   $\langle \text{proof} \rangle$ 

```

```

lemma madd-closed [simp, intro]:  $\llbracket u \in M; v \in M \rrbracket \Longrightarrow u \oplus v \in M$   $\langle \text{proof} \rangle$ 

```

Scaling the zero element gives the zero element.

```

lemma scale-zero-elem:
  assumes a:  $a \in R$ 
  shows  $a \odot \mathbf{0}_M = \mathbf{0}_M$ 
 $\langle \text{proof} \rangle$ 

```

Scaling by the ring zero gives the zero element.

```

lemma scale-zero-scalar:
  assumes v:  $v \in M$ 
  shows  $\mathbf{0} \odot v = \mathbf{0}_M$ 

```

$\langle proof \rangle$

Scaling by the ring negation is the additive inverse of the scaling.

lemma *scale-neg-scalar*:

assumes $a: a \in R$ **and** $v: v \in M$

shows $(- a) \odot v = madd.inverse (a \odot v)$

$\langle proof \rangle$

54.2 Submodules

A *submodule* is a subset containing the zero element and closed under addition and under scaling by arbitrary ring elements.

definition *submodule* :: 'b set $\Rightarrow$ bool

where *submodule* $N \equiv N \subseteq M \wedge \mathbf{0}_M \in N \wedge (\forall u \in N. \forall v \in N. u \oplus v \in N) \wedge (\forall a \in R. \forall v \in N. a \odot v \in N)$

lemma *submoduleI*:

assumes $N \subseteq M$ **and** $\mathbf{0}_M \in N$

and $\bigwedge u v. \llbracket u \in N; v \in N \rrbracket \Longrightarrow u \oplus v \in N$

and $\bigwedge a v. \llbracket a \in R; v \in N \rrbracket \Longrightarrow a \odot v \in N$

shows *submodule* N

$\langle proof \rangle$

lemma *submodule-subset*: *submodule* $N \Longrightarrow N \subseteq M$

$\langle proof \rangle$

lemma *submodule-zero*: *submodule* $N \Longrightarrow \mathbf{0}_M \in N$

$\langle proof \rangle$

lemma *submodule-add*: $\llbracket \text{submodule } N; u \in N; v \in N \rrbracket \Longrightarrow u \oplus v \in N$

$\langle proof \rangle$

lemma *submodule-scale*: $\llbracket \text{submodule } N; a \in R; v \in N \rrbracket \Longrightarrow a \odot v \in N$

$\langle proof \rangle$

A submodule is closed under the module negation (scale by $-\mathbf{1}$).

lemma *submodule-neg*:

assumes N : *submodule* N **and** $v: v \in N$ **shows** *madd.inverse* $v \in N$

$\langle proof \rangle$

The whole module and the trivial submodule are submodules.

lemma *submodule-whole*: *submodule* M

$\langle proof \rangle$

lemma *submodule-trivial*: *submodule* $\{\mathbf{0}_M\}$

$\langle proof \rangle$

The intersection of two submodules is a submodule.

lemma *submodule-inter*:

assumes N : submodule N **and** K : submodule K **shows** submodule $(N \cap K)$
 $\langle proof \rangle$

54.3 The sum of two submodules

The *sum* of two subsets of the module: all sums of an element of each. This is the module analogue of *ideal-sum* in *Chinese-Remainder*, and is the smallest submodule containing both summands when they are themselves submodules.

definition *submodule-sum* :: 'b set $\Rightarrow$ 'b set $\Rightarrow$ 'b set (**infixl** $\langle \oplus_S \rangle$ 65)
where $N \oplus_S K = \{u \oplus v \mid u \in N \wedge v \in K\}$

lemma *submodule-sum-memI*: $\llbracket u \in N; v \in K \rrbracket \Longrightarrow u \oplus v \in N \oplus_S K$
 $\langle proof \rangle$

lemma *submodule-sum-memE*:

assumes $x \in N \oplus_S K$
obtains $u \ v$ **where** $u \in N \ v \in K \ x = u \oplus v$
 $\langle proof \rangle$

The sum contains both summands, since each submodule contains $\mathbf{0}_M$.

lemma *submodule-sum-incl-left*:

assumes N : submodule N **and** K : submodule K **shows** $N \subseteq N \oplus_S K$
 $\langle proof \rangle$

lemma *submodule-sum-incl-right*:

assumes N : submodule N **and** K : submodule K **shows** $K \subseteq N \oplus_S K$
 $\langle proof \rangle$

The sum of two submodules is a submodule. Closure under addition needs commutativity of the module addition to re-pair the four summands; closure under scaling distributes.

theorem *submodule-sum-submodule*:

assumes N : submodule N **and** K : submodule K **shows** submodule $(N \oplus_S K)$
 $\langle proof \rangle$

The sum is commutative, as a set.

lemma *submodule-sum-commute*:

assumes N : submodule N **and** K : submodule K **shows** $N \oplus_S K = K \oplus_S N$
 $\langle proof \rangle$

A trivial intersection makes representations in the submodule sum unique. The existence of such a representation is already exactly *submodule-sum-memE*; this cancellation lemma records the additional content supplied by disjointness.

lemma *submodule-sum-decomposition-unique*:

assumes N : submodule N **and** K : submodule K
and disjoint: $N \cap K = \{\mathbf{0}_M\}$

and $u: u \in N$ **and** $v: v \in K$
and $u': u' \in N$ **and** $v': v' \in K$
and $eq: u \oplus v = u' \oplus v'$
shows $u = u' \wedge v = v'$
 $\langle proof \rangle$

54.4 Linear combinations

A linear combination of a finite set B of module elements with ring coefficients c : the module sum $\bigoplus_{v \in B} c \ v \odot v$.

definition $lincomb :: ('b \Rightarrow 'a) \Rightarrow 'b \ set \Rightarrow 'b$
where $lincomb \ c \ B = madd.fincomp (\lambda v. \ c \ v \odot v) \ B$

lemma $lincomb-closed$:
assumes $B: B \subseteq M$ **and** $c: \bigwedge v. v \in B \implies c \ v \in R$
shows $lincomb \ c \ B \in M$
 $\langle proof \rangle$

lemma $lincomb-empty$ $[simp]$: $lincomb \ c \ \{\} = \mathbf{0}_M$
 $\langle proof \rangle$

The span of a set of module elements: all linear combinations over its finite subsets.

definition $span :: 'b \ set \Rightarrow 'b \ set$
where $span \ S = \{lincomb \ c \ B \mid c \ B. \ finite \ B \wedge B \subseteq S \wedge (\forall v \in B. \ c \ v \in R)\}$

lemma $span-closed$:
assumes $S \subseteq M$ **and** $x \in span \ S$ **shows** $x \in M$
 $\langle proof \rangle$

lemma $span-mono$: $S \subseteq T \implies span \ S \subseteq span \ T$
 $\langle proof \rangle$

end

54.5 A vector space is a module over a field

The module axioms are exactly those of theory *Vector-Space*, so any vector space is a module. (We record the relationship at the locale level; the field's ring reduct supplies the scalar ring.) A dedicated theory could re-derive the theory *Vector-Space* development as the field specialisation of the present one.

end

55 Generators, independence, bases and free modules

theory *Free-Module*

```

imports Module-Basics
begin

```

Over a ring the two halves of “basis” come apart. A spanning set need not be linearly independent, a linearly independent set need not span, and — unlike the vector-space case — a linearly independent spanning set is genuinely stronger than either. So *spanning* and *lin-indep* are defined separately, and a module is *free* when it has a basis, meaning a set that is both.

These definitions are stated for arbitrary — possibly infinite — subsets. Finiteness is not part of either concept; it enters only through the finite index sets inside *lincomb* and *span*. This matters because *madd.fincomp* returns the unit on an infinite index set (*fincomp-infinite*), so a definition that applied *lincomb* directly to an infinite set would silently be talking about the zero of the module rather than about an infinite sum. Quantifying over finite subsets, as *lin-indep* does, avoids that trap; guarding the definition with *finite* would merely hide it.

Vector-Space defines its own finite-dimensional *basis* as bijectivity of the coordinate map, which is the right packaging for the cardinality count there but presumes finiteness. The bridging lemmas relating the two live in that theory, with *finite* as an explicit hypothesis where it does real work rather than baked into the concept.

The uniqueness-of-coordinates theorem *lin-indep-lincomb-unique* below is the reason linear independence is worth isolating: it is what makes a free module the right domain for defining a homomorphism by prescribing values on a basis.

```

context Module
begin

```

55.1 Coefficient functions

A *coefficient function on B* assigns a ring element to each element of *B*. We do not require extensionality: the linear combination *lincomb c B* only ever inspects *c* on *B*, so two coefficient functions agreeing on *B* give the same combination (*lincomb-cong* below).

definition *coeffs-on* :: ('b ⇒ 'a) ⇒ 'b set ⇒ bool
where *coeffs-on c B* ⇔ (∀ v ∈ B. c v ∈ R)

lemma *coeffs-onI*: (∧ v. v ∈ B ⇒ c v ∈ R) ⇒ *coeffs-on c B*
 ⟨proof⟩

lemma *coeffs-onD*: [*coeffs-on c B*; v ∈ B] ⇒ c v ∈ R
 ⟨proof⟩

lemma *coeffs-on-subset*: [*coeffs-on c B*; A ⊆ B] ⇒ *coeffs-on c A*
 ⟨proof⟩

The zero coefficient function.

abbreviation *zero-coeffs* :: 'b ⇒ 'a

where $\text{zero-coeffs} \equiv (\lambda\cdot. \mathbf{0})$

lemma coeffs-on-zero [*simp*, *intro*]: $\text{coeffs-on zero-coeffs } B$
 $\langle \text{proof} \rangle$

A linear combination depends on the coefficients only through their values on the index set.

lemma lincomb-cong :
assumes $B: B \subseteq M$ **and** $\text{eq}: \bigwedge v. v \in B \implies c\ v = d\ v$ **and** $d: \text{coeffs-on } d\ B$
shows $\text{lincomb } c\ B = \text{lincomb } d\ B$
 $\langle \text{proof} \rangle$

The zero combination. Every term is $\mathbf{0} \odot v = \mathbf{0}_M$.

lemma $\text{lincomb-zero-coeffs}$ [*simp*]:
assumes $B: B \subseteq M$ **shows** $\text{lincomb zero-coeffs } B = \mathbf{0}_M$
 $\langle \text{proof} \rangle$

Scaling distributes over a finite module sum. Proved by induction on the index set; kept as a named lemma because inline it is slow enough to hit the proof timeout.

lemma scale-fincomp :
assumes $a: a \in R$ **and** $f: f \in A \rightarrow M$
shows $a \odot \text{madd.fincomp } f\ A = \text{madd.fincomp } (\lambda v. a \odot f\ v)\ A$
 $\langle \text{proof} \rangle$

Enlarging the index set by elements whose coefficient is $\mathbf{0}$ does not change the combination: the extra terms are all $\mathbf{0} \odot v = \mathbf{0}_M$. This is what lets two combinations over different finite index sets be added on their union.

lemma lincomb-mono-zero :
assumes $B: \text{finite } B$ **and** $AB: A \subseteq B$ **and** $BM: B \subseteq M$
and $\text{zero}: \bigwedge v. v \in B - A \implies c\ v = \mathbf{0}$ **and** $c: \text{coeffs-on } c\ B$
shows $\text{lincomb } c\ A = \text{lincomb } c\ B$
 $\langle \text{proof} \rangle$

55.2 Membership in the span

Introduction and elimination for span , in the form the proofs below want.

lemma spanI :
assumes $\text{finite } A$ **and** $A \subseteq S$ **and** $\text{coeffs-on } c\ A$ **and** $x = \text{lincomb } c\ A$
shows $x \in \text{span } S$
 $\langle \text{proof} \rangle$

lemma spanE :
assumes $x \in \text{span } S$
obtains $c\ A$ **where** $\text{finite } A$ $A \subseteq S$ $\text{coeffs-on } c\ A$ $x = \text{lincomb } c\ A$
 $\langle \text{proof} \rangle$

A combination over a singleton is just the scaling. Kept separate: unfolding *lincomb* to *madd.fincomp* inline makes the simplifier work far harder than it needs to (it timed out at 12s in an earlier draft).

lemma *lincomb-singleton* [simp]:
assumes $v: v \in M$ **and** $c: c \cdot v \in R$
shows $\text{lincomb } c \{v\} = c \cdot v \odot v$
 $\langle \text{proof} \rangle$

Each element of a set lies in its span: take the singleton combination with coefficient **1**.

lemma *span-incl*:
assumes $S: S \subseteq M$ **shows** $S \subseteq \text{span } S$
 $\langle \text{proof} \rangle$

Adding coefficient functions pointwise adds the combinations. Both the span-closure proof and the uniqueness proof need this; factored out so each does the *madd.fincomp* manipulation once.

lemma *lincomb-add-coeffs*:
assumes $AM: A \subseteq M$ **and** $c: \text{coeffs-on } c \ A$ **and** $d: \text{coeffs-on } d \ A$
shows $\text{lincomb } (\lambda v. c \cdot v + d \cdot v) \ A = \text{lincomb } c \ A \oplus \text{lincomb } d \ A$
 $\langle \text{proof} \rangle$

The span is a submodule. Closure under addition merges two finite index sets; closure under scaling multiplies through the coefficients.

lemma *span-zero* [simp, intro]: $\mathbf{0}_M \in \text{span } S$
 $\langle \text{proof} \rangle$

lemma *span-madd-closed*:
assumes $S: S \subseteq M$ **and** $x: x \in \text{span } S$ **and** $y: y \in \text{span } S$
shows $x \oplus y \in \text{span } S$
 $\langle \text{proof} \rangle$

lemma *span-scale-closed*:
assumes $S: S \subseteq M$ **and** $a: a \in R$ **and** $x: x \in \text{span } S$
shows $a \odot x \in \text{span } S$
 $\langle \text{proof} \rangle$

theorem *span-submodule*:
assumes $S: S \subseteq M$ **shows** *submodule* ($\text{span } S$)
 $\langle \text{proof} \rangle$

The span is the least submodule containing its generators. This elimination form complements *span-submodule*: any submodule containing S contains every finite linear combination of elements of S .

theorem *span-minimal*:
assumes $N: \text{submodule } N$ **and** $SN: S \subseteq N$
shows $\text{span } S \subseteq N$
 $\langle \text{proof} \rangle$

55.3 Generating sets, independence, bases

S *generates* the module when its span is everything.

definition *spanning* :: 'b set $\Rightarrow$ bool
where *spanning* $S \longleftrightarrow S \subseteq M \wedge \text{span } S = M$

lemma *spanningI*: $\llbracket S \subseteq M; \bigwedge v. v \in M \implies v \in \text{span } S \rrbracket \implies \text{spanning } S$
 $\langle \text{proof} \rangle$

lemma *spanningD*: $\llbracket \text{spanning } S; v \in M \rrbracket \implies v \in \text{span } S$
 $\langle \text{proof} \rangle$

The whole module generates itself.

lemma *spanning-whole*: *spanning* M
 $\langle \text{proof} \rangle$

S is *linearly independent* when no non-trivial finite combination of *distinct* elements vanishes: whenever a finite $A \subseteq S$ has $\text{lincomb } c \ A = \mathbf{0}_M$, all the coefficients on A are $\mathbf{0}$. Over a ring this is the right notion — note it forbids torsion among the generators, which is why not every module is free.

definition *lin-indep* :: 'b set $\Rightarrow$ bool
where *lin-indep* $S \longleftrightarrow S \subseteq M \wedge$
 $(\forall c \ A. \text{finite } A \longrightarrow A \subseteq S \longrightarrow \text{coeffs-on } c \ A \longrightarrow \text{lincomb } c \ A = \mathbf{0}_M \longrightarrow (\forall v \in A. c \ v = \mathbf{0}))$

lemma *lin-indepI*:
 $\llbracket S \subseteq M;$
 $\bigwedge c \ A \ v. \llbracket \text{finite } A; A \subseteq S; \text{coeffs-on } c \ A; \text{lincomb } c \ A = \mathbf{0}_M; v \in A \rrbracket \implies c \ v =$
 $\mathbf{0} \rrbracket$
 $\implies \text{lin-indep } S$
 $\langle \text{proof} \rangle$

lemma *lin-indepD*:
 $\llbracket \text{lin-indep } S; \text{finite } A; A \subseteq S; \text{coeffs-on } c \ A; \text{lincomb } c \ A = \mathbf{0}_M; v \in A \rrbracket \implies c \ v =$
 $\mathbf{0}$
 $\langle \text{proof} \rangle$

lemma *lin-indep-subset*: $\llbracket \text{lin-indep } S; T \subseteq S \rrbracket \implies \text{lin-indep } T$
 $\langle \text{proof} \rangle$

The empty set is linearly independent.

lemma *lin-indep-empty* [*simp, intro*]: *lin-indep* $\{\}$
 $\langle \text{proof} \rangle$

Independence rules out $\mathbf{0}_M$ as a member — the combination $\mathbf{1} \odot \mathbf{0}_M$ vanishes with coefficient $\mathbf{1}$ — but *only over a non-trivial ring*. The hypothesis $\mathbf{1} \neq \mathbf{0}$ is genuinely needed: *Module* extends *Ring*, which does not assume it, and over the trivial ring $\mathbf{1} = \mathbf{0}$ makes every coefficient zero, so $\{\mathbf{0}_M\}$ counts as *lin-indep*.

lemma *lin-indep-not-mzero*:

assumes S : *lin-indep* S **and** nz : $1 \neq 0$ **shows** $0_M \notin S$
 $\langle proof \rangle$

A *basis* is a linearly independent spanning set; a module is *free* when it has one.

definition *module-basis* :: 'b set $\Rightarrow$ bool

where *module-basis* $B \longleftrightarrow$ *spanning* $B \wedge$ *lin-indep* B

definition *free* :: bool

where *free* $\longleftrightarrow (\exists B. \text{module-basis } B)$

lemma *module-basisI*: $\llbracket \text{spanning } B; \text{lin-indep } B \rrbracket \Longrightarrow \text{module-basis } B$

$\langle proof \rangle$

lemma *module-basis-spanning*: *module-basis* $B \Longrightarrow$ *spanning* B

$\langle proof \rangle$

lemma *module-basis-lin-indep*: *module-basis* $B \Longrightarrow$ *lin-indep* B

$\langle proof \rangle$

lemma *freeI*: *module-basis* $B \Longrightarrow$ *free*

$\langle proof \rangle$

The trivial module is free, on the empty basis: its only element 0_M is the empty linear combination, which lies in *span* $\{\}$.

lemma *module-basis-empty-trivial*:

assumes *triv*: $M = \{0_M\}$ **shows** *module-basis* $\{\}$

$\langle proof \rangle$

lemma *free-trivial*:

assumes $M = \{0_M\}$ **shows** *free*

$\langle proof \rangle$

55.4 Uniqueness of coordinates

The point of independence. Over a linearly independent set the coefficients of a linear combination are determined by its value. Stated for a common finite index set A : if two coefficient functions give the same combination, they agree on A .

The proof forms the difference of the two coefficient functions. Its combination is 0_M , so independence forces every difference to vanish, hence $c \ v = d \ v$ by cancellation in the ring's additive group.

theorem *lin-indep-lincomb-unique*:

assumes S : *lin-indep* S **and** A : *finite* $A \subseteq S$

and c : *coeffs-on* $c \ A$ **and** d : *coeffs-on* $d \ A$

and *eq*: *lincomb* $c \ A =$ *lincomb* $d \ A$

and v : $v \in A$

shows $c \ v = d \ v$

<proof>

Consequently, for a basis every module element has *unique* coordinates on any finite index set: existence from generation, uniqueness from independence.

corollary *module-basis-coords-unique:*

assumes *B: module-basis B and A: finite A A ⊆ B*

and *c: coeffs-on c A and d: coeffs-on d A*

and *eq: lincomb c A = lincomb d A*

shows $\forall v \in A. c v = d v$

<proof>

55.5 Coordinates over a basis

For the universal property below we need uniqueness across *different* index sets, not just a common one: two representations of the same element, over possibly different finite subsets of an independent set, must agree once each coefficient function is extended by **0** outside its own index set. The proof pushes both representations onto the union of the two index sets — which changes neither combination, by *lincomb-mono-zero* — and then applies *lin-indep-lincomb-unique* there.

theorem *lin-indep-lincomb-unique-gen:*

assumes *S: lin-indep S*

and *A: finite A A ⊆ S and A': finite A' A' ⊆ S*

and *c: coeffs-on c A and d: coeffs-on d A'*

and *eq: lincomb c A = lincomb d A'*

shows $(\text{if } v \in A \text{ then } c v \text{ else } \mathbf{0}) = (\text{if } v \in A' \text{ then } d v \text{ else } \mathbf{0})$

<proof>

Spans generated by disjoint parts of one independent set meet only at zero. The generating sets themselves need not be finite: span membership supplies the two finite supports on which uniqueness of coordinates is applied.

theorem *span-inter-eq-trivial:*

assumes *S: lin-indep S and AS: A ⊆ S and BS: B ⊆ S*

and *disjoint: A ∩ B = {}*

shows $\text{span } A \cap \text{span } B = \{\mathbf{0}_M\}$

<proof>

Every element of a module with a basis has a representation over some finite subset of it.

lemma *module-basis-repr-exists:*

assumes *B: module-basis B and x: x ∈ M*

obtains *A c where finite A A ⊆ B coeffs-on c A x = lincomb c A*

<proof>

And that representation is unique in the sense of *lin-indep-lincomb-unique-gen*: the coefficients, extended by **0**, are determined by the element.

corollary *module-basis-repr-unique:*

```

assumes B: module-basis B
  and A: finite A  $A \subseteq B$  and A': finite A'  $A' \subseteq B$ 
  and c: coeffs-on c A and d: coeffs-on d A'
  and eq: lincomb c A = lincomb d A'
shows (if  $v \in A$  then c v else 0) = (if  $v \in A'$  then d v else 0)
  <proof>

```

end

end

56 Finite-dimensional vector spaces over a field

```

theory Vector-Space
  imports Free-Module
begin

```

The import of *FiniteProduct* (through *Module*) re-merges HOL's $+/ -$ concrete syntax alongside the ring locale's, which is harmless in term positions (type inference disambiguates) but ambiguous in a locale-header instantiation argument.

```

no-notation plus (infixl <+> 65)
no-notation minus (infixl <-> 65)
unbundle no uminus-syntax

```

A vector space over the field R (with field operations $+$, $\cdot$, **0**, **1**): an abelian group of vectors V under $\oplus$ with zero $\mathbf{0}_V$, together with a scalar multiplication $\odot$ satisfying the module axioms. The purpose of this layer is the structural theorem that a finite field has prime-power order: such a field is a finite-dimensional vector space over its prime subfield, so its cardinality is $|F|^{dim}$.

```

locale Vector-Space = Field +
  fixes vadd :: 'b  $\Rightarrow$  'b  $\Rightarrow$  'b (infixl <⊕> 65)
  and vzero :: 'b (0V)
  and vcarrier :: 'b set (<V>)
  and scale :: 'a  $\Rightarrow$  'b  $\Rightarrow$  'b (infixr <⊙> 75)
assumes vadd-group: Abelian-Group V ( $\oplus$ ) 0V
  and scale-closed:  $\llbracket a \in R; v \in V \rrbracket \Longrightarrow a \odot v \in V$ 
  and scale-distrib-vadd:  $\llbracket a \in R; u \in V; v \in V \rrbracket \Longrightarrow a \odot (u \oplus v) = (a \odot u) \oplus (a \odot v)$ 
  and scale-distrib-add:  $\llbracket a \in R; b \in R; v \in V \rrbracket \Longrightarrow (a + b) \odot v = (a \odot v) \oplus (b \odot v)$ 
  and scale-scale:  $\llbracket a \in R; b \in R; v \in V \rrbracket \Longrightarrow (a \cdot b) \odot v = a \odot (b \odot v)$ 
  and scale-one:  $v \in V \Longrightarrow \mathbf{1} \odot v = v$ 
begin

```

The vector additive group, available through the *vadd* prefix.

```

sublocale vadd: Abelian-Group V ( $\oplus$ ) 0V
  <proof>

```

Every vector space is a module (its scalar ring being a field is the additional constraint). This makes the module-level constructions — *lincomb*, *span*, and the submodule/quotient machinery — available inside the *Vector-Space* locale under the *vadd*-prefixed operations.

sublocale *mod*: *Module* R $(+)$ $(\cdot)$ $\mathbf{0}$ $\mathbf{1}$ $(\oplus)$ $\mathbf{0}_V$ V $(\odot)$
 $\langle \text{proof} \rangle$

Every submodule of a vector space is itself a vector space under the restricted carrier and the same operations. This bridge lets the module-level kernel, image, and quotient constructions reuse the vector-space basis and dimension API.

lemma *submodule-vector-space*:
assumes N : *mod.submodule* N
shows *Vector-Space* R $(+)$ $(\cdot)$ $\mathbf{0}$ $\mathbf{1}$ $(\oplus)$ $\mathbf{0}_V$ N $(\odot)$
 $\langle \text{proof} \rangle$

Basic closure and unit facts.

lemma *vzero-closed* [*simp*, *intro*]: $\mathbf{0}_V \in V$ $\langle \text{proof} \rangle$

lemma *vadd-closed* [*simp*, *intro*]: $\llbracket u \in V; v \in V \rrbracket \implies u \oplus v \in V$
 $\langle \text{proof} \rangle$

lemma *scale-zero-vec*:
assumes $a \in R$
shows $a \odot \mathbf{0}_V = \mathbf{0}_V$
 $\langle \text{proof} \rangle$

Linear combinations and spans come from the module layer. The *mod* sublocale above already supplies *mod.lincomb* and *mod.span* on V , with definitions identical to the ones this theory used to repeat verbatim. We introduce local abbreviations so existing proofs continue to read *lincomb* and *span* unqualified, and re-export the two facts those proofs cite by name.

abbreviation *lincomb* :: $('b \Rightarrow 'a) \Rightarrow 'b \text{ set} \Rightarrow 'b$
where *lincomb* $\equiv$ *mod.lincomb*

abbreviation *span* :: $'b \text{ set} \Rightarrow 'b \text{ set}$
where *span* $\equiv$ *mod.span*

lemmas *lincomb-def* = *mod.lincomb-def*

lemmas *span-def* = *mod.span-def*

lemma *lincomb-closed*:
assumes B : $B \subseteq V$ **and** c : $\bigwedge v. v \in B \implies c \ v \in R$
shows *lincomb* c $B \in V$
 $\langle \text{proof} \rangle$

Spanning and linear independence: the finite-dimensional forms. The module layer defines *mod.spanning* and *mod.lin-indep* for arbitrary subsets (see *Free-Module*). This theory is about the finite-dimensional case — its basis and

dimension results count coordinate vectors — so we keep *finite* variants phrased with a total coefficient function $c \in B \rightarrow_E R$, which is what the coordinate map needs, and relate them to the general notions by the bridging lemmas below.

definition *spanning* :: 'b set $\Rightarrow$ bool

where *spanning* $B \equiv \text{finite } B \wedge B \subseteq V \wedge (\forall v \in V. \exists c \in B \rightarrow_E R. v = \text{lincomb } c \ B)$

definition *lin-indep* :: 'b set $\Rightarrow$ bool

where *lin-indep* $B \equiv \text{finite } B \wedge B \subseteq V \wedge (\forall c \in B \rightarrow_E R. \text{lincomb } c \ B = \mathbf{0}_V \longrightarrow (\forall v \in B. c \ v = \mathbf{0}))$

A set B is a *basis* of V when it is a finite subset whose coordinate map — sending each coefficient vector $c \in B \rightarrow_E R$ to the linear combination $\text{lincomb } c \ B$ — is a bijection onto V . This packages spanning (surjectivity) and linear independence (injectivity) in exactly the form the cardinality count needs.

definition *basis* :: 'b set $\Rightarrow$ bool

where *basis* $B \equiv \text{finite } B \wedge B \subseteq V \wedge \text{bij-betw } (\lambda c. \text{lincomb } c \ B) (B \rightarrow_E R) \ V$

Bridge: the finite forms agree with the module-level notions. On a finite set the two spanning conditions coincide: *mod.spanning* says every vector lies in the span, and on a finite B a span membership is realised by a coefficient function on B itself (extended by $\mathbf{0}$ where the witnessing subset omits an element).

lemma *spanning-iff-mod-spanning*:

assumes *finB*: *finite* B **and** *BV*: $B \subseteq V$

shows *spanning* $B \longleftrightarrow \text{mod.spanning } B$

<proof>

Likewise for independence. The finite form quantifies over total coefficient functions on B ; the module form over coefficient functions on finite subsets. On a finite B these agree: restrict to the subset in one direction, extend by $\mathbf{0}$ in the other.

lemma *lin-indep-iff-mod-lin-indep*:

assumes *finB*: *finite* B **and** *BV*: $B \subseteq V$

shows *lin-indep* $B \longleftrightarrow \text{mod.lin-indep } B$

<proof>

A basis spans: surjectivity of the coordinate map.

lemma *basis-spanning*:

assumes *basis* B **shows** *spanning* B

<proof>

The dimension counting theorem. Over a finite field, a space with a basis of size n has exactly $(\text{card } R)^n$ elements: its coordinate map is a bijection onto the $(\text{card } R)^n$ coefficient vectors $B \rightarrow_E R$.

theorem *card-eq-card-base-pow-dim*:

assumes *basis* B

shows $\text{card } V = \text{card } R \wedge \text{card } B$

<proof>

Over a finite field, any two bases have the same size — so the dimension is well defined. (Both give $\text{card } V = (\text{card } R)^{\text{card } B_i}$, and $2 \leq \text{card } R$ is injective as a base of exponentiation.)

theorem *basis-card-unique-finite*:

assumes *finR*: finite *R* **and** *B1*: basis *B*₁ **and** *B2*: basis *B*₂

shows $\text{card } B_1 = \text{card } B_2$

<proof>

The dimension of the space: the size of a basis (well defined over a finite field).

definition *dimension* :: nat

where *dimension* = card (*SOME B. basis B*)

lemma *dimension-eq*:

assumes *finR*: finite *R* **and** *B*: basis *B* **shows** *dimension* = card *B*

<proof>

end

A vector subspace packages a module-level submodule of a vector space and exposes the induced vector-space structure under the prefix *sub*.

locale *vector-subspace* = *Vector-Space* +

fixes *W* :: 'b set

assumes *W-submodule*: mod.submodule *W*

begin

sublocale *sub*: *Vector-Space* *R* (+) (·) **0** **1** (⊕) **0**_{*V*} *W* (⊙)

<proof>

end

Following the *field-TC* pattern of *Ring-Theory*, we bridge the locale-based *Vector-Space* to Isabelle's type-class arithmetic. A type-class field '*a*' plays the scalar role, and a type-class abelian additive group '*b*' plays the vector role; a scale operation *scale* satisfying the four vector-space axioms then witnesses *Vector-Space* on the whole types as carriers.

lemma *vector-space-TC*:

fixes *scale* :: 'a::field ⇒ 'b::ab-group-add ⇒ 'b

assumes *scale-add-right*: $\bigwedge a \ x \ y. \text{scale } a \ (\text{plus } x \ y) = \text{plus } (\text{scale } a \ x) \ (\text{scale } a \ y)$

and *scale-add-left*: $\bigwedge a \ b \ x. \text{scale } (\text{plus } a \ b) \ x = \text{plus } (\text{scale } a \ x) \ (\text{scale } b \ x)$

and *scale-scale*: $\bigwedge a \ b \ x. \text{scale } a \ (\text{scale } b \ x) = \text{scale } (a * b) \ x$

and *scale-one*: $\bigwedge x. \text{scale } 1 \ x = x$

shows *Vector-Space* (*UNIV* :: 'a set) *plus* (*) **0** **1** *plus 0* (*UNIV* :: 'b set) *scale*

<proof>

Concrete specialisations to Isabelle's built-in *real-vector* and *complex-vector* classes can be added by any consumer.

end

57 The Steinitz exchange lemma and dimension over an arbitrary field

```
theory Steinitz
  imports Vector-Space
begin
```

Restore HOL's arithmetic notation, which *Vector-Space* (via its ancestor *Ring-Theory*) suppresses to disambiguate locale-parameter binding. Steinitz does not bind $+$ or $-$ as locale parameters, so it is safe (and needed for e.g. nat subtraction) to restore them.

```
notation plus (infixl <+> 65)
notation minus (infixl <-> 65)
unbundle uminus-syntax
```

Basis-cardinality uniqueness over a *finite* field was obtained in *Vector-Space* by a counting argument. Here we develop the general (any field, in particular an infinite base such as $\mathbb{Q}$) theory via the *Steinitz exchange lemma*: from a spanning set and a linearly independent set, independent vectors can be exchanged into the spanning set one at a time, whence $|independent| \leq |spanning|$ and any two bases have equal size. It is the linchpin for a general field-extension degree $[K:F]$ and for transitivity of algebraicity.

```
context Vector-Space
begin
```

57.1 Span basics

```
lemma span-closed:
  assumes  $S \subseteq V$  and  $x \in \text{span } S$ 
  shows  $x \in V$ 
  <proof>
```

```
lemma span-mono:
  assumes  $S \subseteq T$  shows  $\text{span } S \subseteq \text{span } T$ 
  <proof>
```

Every vector of S lies in its span.

```
lemma span-incl:
  assumes  $S \subseteq V$   $v \in S$ 
  shows  $v \in \text{span } S$ 
  <proof>
```

```
lemma span-subset-V:
  assumes  $S \subseteq V$  shows  $\text{span } S \subseteq V$ 
  <proof>
```

57.2 Extension of a linear combination by zero coefficients

Scaling by the field zero gives the zero vector: from $(\mathbf{0} + \mathbf{0}) \odot v = \mathbf{0} \odot v \oplus \mathbf{0} \odot v$ and cancellation.

lemmas *scale-zero-scalar* = *mod.scale-zero-scalar*

Extending a linear combination's support by vectors with zero coefficient leaves it unchanged: the extra terms $\mathbf{0} \odot v = \mathbf{0}_V$ vanish in the vector sum.

lemma *lincomb-extend-zero*:

assumes $B: B \subseteq V$ **and** $C: \text{finite } C \subseteq V$ **and** $c: \bigwedge v. v \in B \implies c v \in R$
and $\text{disj}: B \cap C = \{\}$ **and** $\text{czero}: \bigwedge v. v \in C \implies c v = \mathbf{0}$
and $\text{finB}: \text{finite } B$
shows $\text{lincomb } c (B \cup C) = \text{lincomb } c B$

<proof>

57.3 Span is a subspace

Over a common finite support B , linear combinations add coefficientwise: $\text{lincomb } c B \oplus \text{lincomb } d B = \text{lincomb } (\lambda v. c v + d v) B$ (by *vadd.fincomp-comp* and *scale-distrib-add*).

lemma *lincomb-add*:

assumes $B: B \subseteq V$ **and** $\text{finB}: \text{finite } B$
and $c: \bigwedge v. v \in B \implies c v \in R$ **and** $d: \bigwedge v. v \in B \implies d v \in R$
shows $\text{lincomb } c B \oplus \text{lincomb } d B = \text{lincomb } (\lambda v. c v + d v) B$

<proof>

A scalar pulls through a vector finite sum: $a \odot \text{vadd.fincomp } g A = \text{vadd.fincomp } g (a \odot A)$ (by *scale-distrib-vadd*, induction on A).

lemmas *scale-fincomp* = *mod.scale-fincomp*

Scaling a linear combination scales its coefficients: $a \odot \text{lincomb } c B = \text{lincomb } (\lambda v. a \cdot c v) B$.

lemma *lincomb-scale*:

assumes $B: B \subseteq V$ **and** $\text{finB}: \text{finite } B$ **and** $a: a \in R$
and $c: \bigwedge v. v \in B \implies c v \in R$
shows $a \odot \text{lincomb } c B = \text{lincomb } (\lambda v. a \cdot c v) B$

<proof>

57.4 Span closure, idempotence and transitivity

Two coefficient functions agreeing on the support give the same linear combination.

lemma *lincomb-cong*:

assumes $B \subseteq V$ **and** $\bigwedge v. v \in B \implies d v \in R$ **and** $\bigwedge v. v \in B \implies c v = d v$
shows $\text{lincomb } c B = \text{lincomb } d B$

<proof>

Enlarging the support of a linear combination by vectors whose coefficient is $\mathbf{0}$ leaves its value unchanged — the general (non-disjoint) form of *lincomb-extend-zero*, obtained by taking the extra part to be the vectors of D outside B .

lemma *lincomb-zero-extend-eq*:

assumes $B: B \subseteq V$ **and** $D: \text{finite } D \ D \subseteq V$ **and** $BD: B \subseteq D$
and $c: \bigwedge v. v \in B \implies c \ v \in R$ **and** $czero: \bigwedge v. [v \in D; v \notin B] \implies c \ v = \mathbf{0}$
shows $\text{lincomb } c \ D = \text{lincomb } c \ B$

<proof>

The zero vector is the empty linear combination, hence lies in every span.

lemma *span-zero*: $\mathbf{0}_V \in \text{span } S$

<proof>

A span is closed under scalar multiplication.

lemmas *span-scale* = *mod.span-scale-closed*

A span is closed under vector addition.

lemmas *span-vadd* = *mod.span-madd-closed*

Consequently a linear combination of vectors *drawn from a span* stays in the span: induction on the support, using *span-zero*, *span-scale*, *span-vadd*.

lemma *span-lincomb-closed*:

assumes $S: S \subseteq V$ **and** $\text{fin}B: \text{finite } B$ **and** $BS: B \subseteq \text{span } S$
and $cR: \bigwedge v. v \in B \implies c \ v \in R$
shows $\text{lincomb } c \ B \in \text{span } S$

<proof>

Span is idempotent, so it is a genuine closure operator.

lemma *span-span*:

assumes $S: S \subseteq V$ **shows** $\text{span } (\text{span } S) = \text{span } S$

<proof>

Transitivity: if S lies inside the span of T , its span does too.

lemma *span-trans*:

assumes $ST: S \subseteq \text{span } T$ **and** $T: T \subseteq V$

shows $\text{span } S \subseteq \text{span } T$

<proof>

The membership form of transitivity, matching the shape used in the exchange lemma: absorbing a spanned vector into the generating set does not enlarge the span.

lemma *span-trans-mem*:

assumes $S \subseteq V$ **and** $x \in \text{span } S$ **and** $y \in \text{span } (\text{insert } x \ S)$

shows $y \in \text{span } S$

<proof>

57.5 Vector subtraction inside a span

Negating a scalar coefficient negates the scaled vector: $(-a) \odot v$ is the vector-group inverse of $a \odot v$. From $(a - a) \odot v = \mathbf{0}_V$.

lemmas *scale-neg-scalar* = *mod.scale-neg-scalar*

A span is closed under vector negation.

lemma *span-neg*:

assumes $S \subseteq V$ **and** $x \in \text{span } S$

shows $\text{vadd.inverse } x \in \text{span } S$

<proof>

A span is closed under vector subtraction.

lemma *span-diff*:

assumes $S: S \subseteq V$ **and** $x: x \in \text{span } S$ **and** $y: y \in \text{span } S$

shows $x \oplus \text{vadd.inverse } y \in \text{span } S$

<proof>

57.6 The exchange step

Exchange step. If a lies in the span of *insert* b S but not in the span of S , then b lies in the span of *insert* a S — so b may be swapped out for a .

lemma *in-span-insert*:

assumes $S: S \subseteq V$ **and** $b: b \in V$

and $a: a \in \text{span } (\text{insert } b \text{ } S)$ **and** $na: a \notin \text{span } S$

shows $b \in \text{span } (\text{insert } a \text{ } S)$

<proof>

Delete form. If a is spanned by S but not by $S - \{b\}$, then b can be exchanged into the generating set in place of a .

lemma *in-span-delete*:

assumes $S: S \subseteq V$ **and** $b: b \in V$

and $a: a \in \text{span } S$ **and** $na: a \notin \text{span } (S \setminus \{b\})$

shows $b \in \text{span } (\text{insert } a \text{ } (S \setminus \{b\}))$

<proof>

57.7 Independence as irredundancy

The geometric reading of linear independence: no member of an independent set lies in the span of the others. This is the bridge from the coordinate definition *lin-indep* to the span-theoretic exchange arguments. If some $w \in B$ were in $\text{span } (B - \{w\})$, its witnessing combination, moved to one side with coefficient -1 on w , would be a nontrivial dependence.

lemma *lin-indep-not-in-span*:

assumes *ind*: *lin-indep* B **and** $w: w \in B$

shows $w \notin \text{span } (B \setminus \{w\})$

<proof>

Conversely, over a field, a finite set none of whose members lies in the span of the others is linearly independent. Given a dependence $\text{lincomb } c \ B = \mathbf{0}_V$ with some $c \ w \neq \mathbf{0}$, solving for w exhibits w in the span of the rest.

lemma *not-in-span-lin-indep*:

assumes *finB*: finite B **and** $B \subseteq V$
and *irred*: $\bigwedge w. w \in B \implies w \notin \text{span } (B \setminus \{w\})$

shows *lin-indep* B

<proof>

Adjoining a vector outside the span preserves linear independence.

lemma *lin-indep-insert*:

assumes *ind*: *lin-indep* A **and** $x: x \in V$ **and** *outside*: $x \notin \text{span } A$

shows *lin-indep* (*insert* $x \ A$)

<proof>

57.8 Bases span and are independent

A spanning set spans the whole space: every vector of V lies in its span.

lemma *spanning-span-all*:

assumes *spanning* $B \ v \in V$

shows $v \in \text{span } B$

<proof>

A basis is linearly independent: the coordinate map is injective, and the zero vector is *lincomb mod.zero-coeffs* B , so any coordinate vector mapping to $\mathbf{0}_V$ must be the zero one.

lemma *basis-lin-indep*:

assumes B : *basis* B

shows *lin-indep* B

<proof>

A finite vector-space basis is also a basis in the underlying free-module sense. This bridge lets constructions defined by the universal property of free modules consume the coordinate-map basis used by the dimension theory.

lemma *basis-module-basis*:

assumes B : *basis* B

shows *mod.module-basis* B

<proof>

Spanning forces equality with a containing independent set. If $T \subseteq S$, S is independent and $S \subseteq \text{span } T$, then $S = T$: any $w \in S - T$ would lie in $\text{span } T \subseteq \text{span } (S - \{w\})$, contradicting independence.

lemma *spanning-subset-independent*:

assumes TS : $T \subseteq S$ **and** iS : *lin-indep* S **and** SsT : $S \subseteq \text{span } T$

shows $S = T$

<proof>

57.9 The Steinitz exchange lemma

An independent set contained in the span of a set T is contained in the span of an equinumerous “exchanged” set drawn from $S \cup T$: independent vectors are traded into T one at a time. Induction on $\text{card } (T - S)$, following Steinitz.

lemma *exchange-lemma*:

assumes *finite* T **and** *lin-indep* S **and** $S \subseteq \text{span } T$ **and** $T \subseteq V$

shows $\exists T'. \text{card } T' = \text{card } T \wedge \text{finite } T' \wedge S \subseteq T' \wedge T' \subseteq S \cup T \wedge S \subseteq \text{span } T'$

<proof>

57.10 The cardinality bound and uniqueness of dimension

Steinitz bound. A linearly independent set is no larger than any set whose span contains it — in particular, than any spanning set.

theorem *independent-le-span*:

assumes *iS*: *lin-indep* S **and** *fT*: *finite* T **and** *TV*: $T \subseteq V$ **and** *spST*: $S \subseteq \text{span } T$

shows $\text{card } S \leq \text{card } T$

<proof>

Uniqueness of dimension over an arbitrary field. Any two bases of V have the same cardinality: each is independent and spans, so each bounds the other by the Steinitz bound. This generalises *basis-card-unique-finite*, dropping the finiteness of the base field.

theorem *basis-card-unique*:

assumes *B1*: *basis* B_1 **and** *B2*: *basis* B_2

shows $\text{card } B_1 = \text{card } B_2$

<proof>

Consequently the *dimension* is the size of *any* basis, over an arbitrary field — the finite-field hypothesis of *dimension-eq* is no longer needed.

theorem *dimension-eq-any-field*:

assumes *B*: *basis* B **shows** $\text{dimension} = \text{card } B$

<proof>

57.11 Existence of a basis

basis is defined as bijectivity of the coordinate map, and *basis-spanning* and *basis-lin-indep* take it apart. The converse — that spanning together with independence *gives* a basis — is what one needs in order to *build* a basis, and it is the missing introduction rule: surjectivity of the coordinate map is spanning, and injectivity is independence, applied to the difference of two coordinate vectors.

lemma *basisI*:

assumes *sp*: *spanning* B **and** *ind*: *lin-indep* B

shows *basis* B

<proof>

The zero vector space has the empty basis. This small introduction rule is useful when dimension arguments reduce a kernel or intersection to the trivial subspace.

lemma *basis-empty-trivial*:

assumes *triv*: $V = \{0_V\}$

shows *basis* $\{\}$

<proof>

Basis extension. Every finite independent set extends to a basis inside its union with any given basis. At each step, a basis vector outside the current span is adjoined; the finite difference from the given basis is the termination measure.

theorem *basis-extension*:

assumes *ind*: *lin-indep* A **and** B : *basis* B

shows $\exists C. A \subseteq C \wedge C \subseteq A \cup B \wedge$ *basis* C

<proof>

Every finitely spanned space has a basis, extracted from any finite spanning set. Discard redundant vectors one at a time: if some $w \in B$ lies in the span of the others then $B - \{w\}$ still spans (by *span-trans-mem*), and it is strictly smaller, so the process terminates. What it terminates at is a set no member of which is spanned by the rest, which by *not-in-span-lin-indep* is exactly independence. This is the result that turns “finite-dimensional” into “has a basis”, and hence the one that lets the dimension theory above be applied to a space presented by generators.

theorem *basis-exists-from-spanning*:

assumes *spanning* B

shows $\exists B' \subseteq B.$ *basis* B'

<proof>

A finite vector space has cardinality $(\text{card } R)^{\text{dimension}}$.

theorem *card-eq-card-base-pow-dimension*:

assumes *finV*: *finite* V

shows $\text{card } V = \text{card } R \wedge \text{dimension}$

<proof>

A uniform finite bound on the cardinalities of independent sets already gives a basis. Choose an independent set of maximum cardinality; if a vector lay outside its span, adjoining that vector would produce a strictly larger independent set.

theorem *basis-exists-of-independent-card-bound*:

assumes *bound*: $\bigwedge A. \text{lin-indep } A \implies \text{card } A \leq n$

shows $\exists A.$ *basis* A

<proof>

Hence the dimension of a finitely spanned space is bounded by any spanning set.

corollary *dimension-le-spanning*:

assumes *spanning B* **shows** $\text{dimension} \leq \text{card } B$
 $\langle \text{proof} \rangle$

end

context *vector-subspace*
begin

Linear combinations and spans do not depend on whether their vectors are viewed in the ambient space or in the subspace; only the carrier side-condition changes.

lemma *sub-fincomp-eq*:
assumes *finA: finite A* **and** *f: f ∈ A → W*
shows *sub.vadd.fincomp f A = vadd.fincomp f A*
 $\langle \text{proof} \rangle$

lemma *sub-lincomb-eq*:
assumes *finA: finite A* **and** *AW: A ⊆ W*
and *c: ⋀ v. v ∈ A ⟹ c v ∈ R*
shows *sub.lincomb c A = lincomb c A*
 $\langle \text{proof} \rangle$

lemma *sub-span-eq*:
assumes *AW: A ⊆ W*
shows *sub.span A = span A*
 $\langle \text{proof} \rangle$

lemma *sub-lin-indep-iff*:
assumes *AW: A ⊆ W*
shows *sub.lin-indep A ⟷ lin-indep A*
 $\langle \text{proof} \rangle$

Subspaces of finite-dimensional spaces have bases. The possible cardinalities of independent subsets of W are bounded by the size of an ambient basis. Choose an independent set of maximum cardinality; adjoining any vector outside its span would contradict maximality, so it spans the subspace.

theorem *subspace-basis-exists*:
assumes *B: basis B*
shows $\exists A. \text{sub.basis } A$
 $\langle \text{proof} \rangle$

A subspace of a finite-dimensional space cannot have larger dimension than the ambient space. This is the set-based counterpart of the standard monotonicity theorem for dimension: a subspace basis is also ambiently independent, so Steinitz bounds it by any ambient basis.

theorem *subspace-dimension-le*:
assumes *B: basis B*
shows *sub.dimension ≤ dimension*

<proof>

In finite dimension, a subspace has full dimension exactly when it is the whole ambient space. For the non-trivial direction, extend a subspace basis to an ambient basis; equality of dimensions leaves no additional basis vectors.

theorem *subspace-eq-ambient-iff-dimension-eq*:
 assumes *B*: basis *B*
 shows $W = V \longleftrightarrow \text{sub.dimension} = \text{dimension}$
<proof>

The only zero-dimensional subspace is the trivial one. The ambient basis assumption supplies a basis of the subspace; dimension zero then forces that basis to be empty.

theorem *subspace-dimension-eq-zero-iff*:
 assumes *B*: basis *B*
 shows $\text{sub.dimension} = 0 \longleftrightarrow W = \{0_V\}$
<proof>

end

end

58 A finite subgroup of the multiplicative group of a field is cyclic

theory *Field-Mult-Cyclic*
 imports *Poly-Ring Group-Operations*
begin

In particular the multiplicative group of a finite field (e.g. $GF(p^n)$) is cyclic. The argument splits into a group-theoretic part and a field-theoretic part. If a finite group G satisfies the *root bound* that for every $0 < n$ at most n elements x satisfy $x^n = 1$, then G is cyclic. The field part supplies the root bound: the elements x of the multiplicative group with $x^n = 1$ are roots of the polynomial $X^n - 1$, and a field has at most n such roots.

58.1 The order of a power in a finite group

context *Group*
begin

In a finite group, the order of *local.power a k* is *element-order a div gcd (element-order a) k*. This is the key computation behind counting the elements of each order in a cyclic group.

lemma *element-order-power*:
 assumes *fin*: finite *G* **and** *aG*: $a \in G$

shows $\text{element-order } (\text{power } a \ k) = \text{element-order } a \ \text{div } \text{gcd } (\text{element-order } a) \ k$
 $\langle \text{proof} \rangle$

end

58.2 Counting elements of each order in a cyclic group

context *Group*

begin

The elements of order exactly e in the cyclic group generated by a (where a has order e) are exactly its generators: the powers $\text{local.power } a \ k$ with k below e and coprime to it.

lemma *card-order-e-in-cyclic*:

assumes $\text{fin: finite } G$ **and** $aG: a \in G$ **and** $\text{ord-a: element-order } a = e$

shows $\text{card } \{x \in \text{cyclic-subgroup } a. \text{element-order } x = e\} = \text{card } \{k. k < e \wedge \text{gcd } e \ k = 1\}$

$\langle \text{proof} \rangle$

end

58.3 Counting the Residues Coprime to a Modulus

The cyclicity criterion for a finite group needs two arithmetic facts about the number of residues below e coprime to e : that it is positive, and that summing it over the divisors of d recovers d .

lemma *coprime-count-pos*:

fixes $e :: \text{nat}$

assumes $0 < e$ **shows** $0 < \text{card } \{k. k < e \wedge \text{gcd } e \ k = 1\}$

$\langle \text{proof} \rangle$

The residues below d whose greatest common divisor with d is exactly f are the multiples $j * f$ of f by residues below $d \ \text{div } f$ coprime to it.

lemma *card-gcd-class*:

fixes $d \ f :: \text{nat}$

assumes $d: 0 < d$ **and** $f: f \ \text{dvd } d$

shows $\text{card } \{j. j < d \ \text{div } f \wedge \text{gcd } (d \ \text{div } f) \ j = 1\} = \text{card } \{k. k < d \wedge \text{gcd } k \ d = f\}$

$\langle \text{proof} \rangle$

Summing the coprime counts over the divisors of d recovers d : the residues below d are partitioned by their greatest common divisor with d .

theorem *coprime-count-divisor-sum*:

fixes $d :: \text{nat}$

assumes $d: 0 < d$

shows $(\sum e \mid e \ \text{dvd } d. \text{card } \{k. k < e \wedge \text{gcd } e \ k = 1\}) = d$

$\langle \text{proof} \rangle$

58.4 The abstract group criterion for being cyclic

context *Group*
begin

The set of elements of order exactly e in a finite group.

Every element of a finite group has order dividing $\text{card } G$, so the group is partitioned by order over the divisors of $\text{card } G$.

lemma *card-G-sum-orders*:
assumes $\text{fin}: \text{finite } G$
shows $\text{card } G = (\sum e \mid e \text{ dvd } \text{card } G. \text{card } \{x \in G. \text{element-order } x = e\})$
 $\langle \text{proof} \rangle$

The root bound: for each positive n , at most n elements satisfy $x^n = 1$. Under this hypothesis, at most $\text{card } \{k. k < e \wedge \text{gcd } e \ k = 1\}$ elements have order exactly e .

lemma *card-order-e-le-coprime-count*:
assumes $\text{fin}: \text{finite } G$
and bound: $\bigwedge n. n > 0 \implies \text{card } \{x \in G. \text{power } x \ n = 1\} \leq n$
and e-pos: $e > 0$
shows $\text{card } \{x \in G. \text{element-order } x = e\} \leq \text{card } \{k. k < e \wedge \text{gcd } e \ k = 1\}$
 $\langle \text{proof} \rangle$

The abstract criterion: a finite group satisfying the root bound is cyclic.

theorem *cyclic-of-root-bound*:
assumes $\text{fin}: \text{finite } G$ **and** $\text{ne}: G \neq \{\}$
and bound: $\bigwedge n. n > 0 \implies \text{card } \{x \in G. \text{power } x \ n = 1\} \leq n$
shows $\exists g \in G. \text{cyclic-subgroup } g = G$
 $\langle \text{proof} \rangle$

end

58.5 The multiplicative group of a field and the root bound

context *Field*
begin

The nonzero elements of the field.

definition $Fstar :: 'a \text{ set}$
where $Fstar = R - \{0\}$

lemma *Fstar-iff [simp]*: $x \in Fstar \longleftrightarrow x \in R \wedge x \neq 0$
 $\langle \text{proof} \rangle$

lemma *one-in-Fstar*: $1 \in Fstar$
 $\langle \text{proof} \rangle$

The nonzero elements form a group under multiplication.

theorem *Group-Fstar: Group Fstar* $(\cdot)$ **1**
 $\langle proof \rangle$

end

58.6 The polynomial root bound, via the polynomial ring

The number of n th roots of unity in a field is at most n . Rather than reproving a bound on polynomial roots by hand, we reuse *Field.card-roots-le-degree* from *Poly-Ring*: the n th roots of unity are exactly the roots of the polynomial $X^n - 1$, which is nonzero of degree n .

context *Field*
begin

The ring power *rpow* of *Poly-Ring* agrees with the multiplicative group power.

lemma *rpow-eq-power*: *rpow* x n = *multiplicative.power* x n
 $\langle proof \rangle$

The polynomial $X^n - 1$, as a coefficient sequence: -1 at 0 , 1 at n , and 0 elsewhere (valid for $0 < n$).

definition *xn-minus-1* :: $\text{nat} \Rightarrow (\text{nat} \Rightarrow 'a)$
where *xn-minus-1* $n \equiv (\lambda i. \text{if } i = 0 \text{ then } -1 \text{ else if } i = n \text{ then } 1 \text{ else } 0)$

lemma *xn-minus-1-closed*: *xn-minus-1* $n \in \text{poly-carrier}$
 $\langle proof \rangle$

lemma *xn-minus-1-coeff*: *xn-minus-1* n $i \in R$
 $\langle proof \rangle$

lemma *degree-xn-minus-1*:
assumes $n: n > 0$ **shows** *degree* (*xn-minus-1* n) = n
 $\langle proof \rangle$

lemma *xn-minus-1-nonzero*:
assumes $n: n > 0$ **shows** *xn-minus-1* $n \neq 0_P$
 $\langle proof \rangle$

Its value at x is *multiplicative.power* x $n - 1$.

lemma *eval-xn-minus-1*:
assumes $n: n > 0$ **and** $x: x \in R$
shows *eval* x (*xn-minus-1* n) = *multiplicative.power* x $n + (-1)$
 $\langle proof \rangle$

The roots of unity are exactly the roots of $X^n - 1$.

lemma *card-nth-roots-of-unity*:
assumes $n: n > 0$
shows *finite* $\{x \in R. \text{multiplicative.power } x \text{ } n = 1\}$

$\wedge \text{card } \{x \in R. \text{multiplicative.power } x \ n = \mathbf{1}\} \leq n$
 $\langle \text{proof} \rangle$

end

58.7 The main theorem

The *Fstar*-power (multiplicative power in the group of nonzero elements) coincides with the field's multiplicative power, so the root bound transfers to the group.

context *Field*
begin

lemma *Fstar-power-eq*:
Monoid.power $(\cdot) \ \mathbf{1} \ x \ n = \text{multiplicative.power } x \ n$
 $\langle \text{proof} \rangle$

Any finite subgroup of the multiplicative group of a field is cyclic.

theorem *finite-subgroup-Fstar-cyclic*:
assumes *sub*: *Subgroup* *H Fstar* $(\cdot) \ \mathbf{1}$ **and** *fin*: *finite* *H*
shows $\exists g \in H. \text{Group.cyclic-subgroup } (\cdot) \ \mathbf{1} \ g = H$
 $\langle \text{proof} \rangle$

In particular, if the whole multiplicative group *Fstar* is finite (e.g. a finite field), it is cyclic.

corollary *finite-field-mult-cyclic*:
assumes *fin*: *finite* *Fstar*
shows $\exists g \in \text{Fstar}. \text{Group.cyclic-subgroup } (\cdot) \ \mathbf{1} \ g = \text{Fstar}$
 $\langle \text{proof} \rangle$

end

end

59 A field extension as a vector space over a subfield

theory *Extension-Vector-Space*
imports *Steinitz Ring-Theory Subfield*
begin

We feed the type-class field operations into the abstract *Vector-Space*. For subfields $K \subseteq L$ of a type-class field, *L* is a vector space over *K*: the vectors are the elements of *L*, addition and the zero are the field's, and scalar multiplication is field multiplication (restricted to scalars from *K*). This is the linchpin

that carries the Steinitz results — *Vector-Space.span*, *Vector-Space.basis*, *Vector-Space.dimension* and *Vector-Space.basis-card-unique* — over to field extensions, giving a general $[L:K]$ and its tower law.

notation *plus* (**infixl** $\langle + \rangle$ 65)
notation *minus* (**infixl** $\langle - \rangle$ 65)
unbundle *uminus-syntax*

59.1 A subfield is a locale-based field

The carrier K of a *Subfield* satisfies the locale-based *Field* axioms under the ambient type-class operations. Built bottom-up exactly as in *Ring-Theory*, but with carrier K in place of the whole type.

context *Subfield*
begin

lemma *sf-add-group*: *Group* K $(+)$ 0
 $\langle proof \rangle$

lemma *sf-add-abelian*: *Abelian-Group* K $(+)$ 0
 $\langle proof \rangle$

lemma *sf-mult-cmonoid*: *commutative-monoid* K $(*)$ 1
 $\langle proof \rangle$

lemma *sf-ring*: *Ring* K $(+)$ $(*)$ 0 1
 $\langle proof \rangle$

lemma *sf-commutative-ring*: *commutative-ring* K $(+)$ $(*)$ 0 1
 $\langle proof \rangle$

lemma *sf-field*: *Field* K $(+)$ $(*)$ 0 1
 $\langle proof \rangle$

end

59.2 The extension as a vector space

Two nested subfields $K \subseteq L$. The larger field L is the space of vectors, the smaller field K supplies the scalars.

locale *subfield-tower* = *base*: *Subfield* K + *ext*: *Subfield* L
for K L :: '*a* :: *field set* +
assumes *base-subset*: $K \subseteq L$
begin

L is a K -vector space under the ambient field operations. (The locale name is qualified: *HOL.Vector-Spaces* is imported transitively and also defines a *vector-space* locale.)

sublocale *vs*: *Vector-Space* $K (+) (*) 0 1 (+) 0 L (*)$
 ⟨*proof*⟩

end

context *subfield-tower*
begin

Over the extension, the abstract additive *commutative-monoid.fincomp* of the vector group is just the ordinary finite sum *sum*, since *vadd* = (+) and the summands lie in *L*.

lemma *vadd-fincomp-eq-sum*:
assumes *finite B and* $f \in B \rightarrow L$
shows *vs.vadd.fincomp f B = sum f B*
 ⟨*proof*⟩

Hence a *Vector-Space.lincomb* over a finite index set of vectors is the field sum of the scaled vectors.

lemma *vs-lincomb-eq-sum*:
assumes *finite B and* $B \subseteq L$ **and** $\bigwedge v. v \in B \implies c \ v \in K$
shows *vs.lincomb c B =* $(\sum_{v \in B. c \ v * v})$
 ⟨*proof*⟩

A finite K-scaled sum of vectors in L lies in the span of those vectors.

lemma *sum-scale-in-span*:
assumes *finite A and* $\bigwedge x. x \in A \implies g \ x \in L$ **and** $\bigwedge x. x \in A \implies f \ x \in K$
shows $(\sum_{x \in A. f \ x * g \ x) \in vs.span (g \ ` A)$
 ⟨*proof*⟩

Specialise the finite vector-space cardinality bridge to a finite field extension. The carrier of the larger field is the vector-space carrier, while *vs.dimension* is the extension degree in the existing tower interpretation.

corollary *finite-subfield-tower-cardinality*:
assumes *fnL: finite L*
shows *card L = card K ^ vs.dimension*
 ⟨*proof*⟩

Basis extension is part of the field-extension API as well: a finite independent set of vectors can be completed inside its union with an existing basis.

corollary *subfield-tower-basis-extension*:
assumes *ind: vs.lin-indep A and B: vs.basis B*
shows $\exists C. A \subseteq C \wedge C \subseteq A \cup B \wedge vs.basis \ C$
 ⟨*proof*⟩

A finite extension has a basis in the same interpretation used for its dimension. This re-exports the basis-extraction theorem at the field-extension boundary, so clients do not have to reconstruct the finite spanning argument through the module implementation.

corollary *finite-subfield-tower-basis-exists:*

assumes *finL: finite L*

shows $\exists B. \text{vs.basis } B$

<proof>

More generally, a finite spanning set for the larger field can be reduced to a basis. This is the reusable bridge for extensions presented by generators rather than by a finite carrier.

corollary *subfield-tower-basis-exists-of-finite-spanning:*

assumes *finB: finite B and BL: $B \subseteq L$ and span: $L \subseteq \text{vs.span } B$*

shows $\exists C. C \subseteq B \wedge \text{vs.basis } C$

<proof>

The dimension of a nontrivial finite field extension is positive. Stating this once avoids repeated cardinality arguments when a client needs a genuine extension degree.

corollary *finite-subfield-tower-dimension-pos:*

assumes *finL: finite L*

shows *vs.dimension* > 0

<proof>

end

Every subfield sits in the trivial tower over itself, and every pair of nested subfields forms a tower; the sublocale then equips it with the full Steinitz vocabulary.

lemma *subfield-tower-refl:*

assumes *Subfield K* **shows** *subfield-tower K K*

<proof>

end

60 Frobenius on finite subfields

theory *Finite-Field-Frobenius*

imports *Prime-Subfield Steinitz Field-Mult-Cyclic Extension-Vector-Space*

begin

60.1 Cardinality of finite fields

context *Field*

begin

A finite field is a finite-dimensional vector space over its canonical prime subfield. Taking the whole carrier as an initial finite spanning set supplies a basis; the coordinate bijection then counts the field.

theorem *finite-field-cardinality:*

assumes *fin*: *finite R*
shows $\exists n > 0. \text{card } R = \text{characteristic } \wedge n$
 $\langle \text{proof} \rangle$

corollary *finite-field-cardinality-prime-power*:

assumes *fin*: *finite R*
shows $\exists p \ n. \text{prime } p \wedge n > 0 \wedge \text{card } R = p \wedge n$
 $\langle \text{proof} \rangle$

corollary *characteristic-dvd-card-finite-field*:

assumes *fin*: *finite R*
shows *characteristic dvd card R*
 $\langle \text{proof} \rangle$

end

A finite subfield is represented here as a carrier set inside an ambient type-class field. The existing finite-field cardinality theorem therefore speaks about the characteristic of the carrier-set locale. The first lemmas identify those two presentations; the remaining results expose Frobenius without introducing a second field representation.

definition *frobenius-power* :: *nat* $\Rightarrow$ *'a* :: *monoid-mult* $\Rightarrow$ *'a*
where *frobenius-power* *q x* = *x* $\wedge$ *q*

context *Subfield*

begin

interpretation *carrier*: *Field K* (+) (*) 0 1
 $\langle \text{proof} \rangle$

lemma *carrier-natmult-eq-of-nat*:

carrier.natmult n = *of-nat n*
 $\langle \text{proof} \rangle$

lemma *carrier-characteristic-eq-CHAR*:

assumes *fin*: *finite K*
shows *carrier.characteristic* = *CHAR('a)*
 $\langle \text{proof} \rangle$

theorem *finite-subfield-cardinality-char-power*:

assumes *fin*: *finite K*
shows $\exists n > 0. \text{card } K = \text{CHAR}('a) \wedge n$
 $\langle \text{proof} \rangle$

lemma *finite-subfield-CHAR-prime*:

assumes *fin*: *finite K*
shows *prime CHAR('a)*
 $\langle \text{proof} \rangle$

end

lemma *frobenius-power-add-char-power*:

fixes $x\ y :: 'a :: \text{field}$

assumes $\text{char}: \text{prime } \text{CHAR}('a)$ **and** $q: q = \text{CHAR}('a) \wedge n$

shows $\text{frobenius-power } q\ (x + y) = \text{frobenius-power } q\ x + \text{frobenius-power } q\ y$

$\langle \text{proof} \rangle$

lemma *frobenius-power-mult*:

fixes $x\ y :: 'a :: \text{comm-monoid-mult}$

shows $\text{frobenius-power } q\ (x * y) = \text{frobenius-power } q\ x * \text{frobenius-power } q\ y$

$\langle \text{proof} \rangle$

lemma *frobenius-power-one* [simp]:

$\text{frobenius-power } q\ (1 :: 'a :: \text{monoid-mult}) = 1$

$\langle \text{proof} \rangle$

lemma *frobenius-power-zero* [simp]:

assumes $q > 0$

shows $\text{frobenius-power } q\ (0 :: 'a :: \text{semiring-1}) = 0$

$\langle \text{proof} \rangle$

lemma *frobenius-power-uminus-char-power*:

fixes $x :: 'a :: \text{field}$

assumes $\text{char}: \text{prime } \text{CHAR}('a)$ **and** $q: q = \text{CHAR}('a) \wedge n$

shows $\text{frobenius-power } q\ (-x) = -(\text{frobenius-power } q\ x)$

$\langle \text{proof} \rangle$

lemma *frobenius-power-inj*:

fixes $x\ y :: 'a :: \text{field}$

assumes $\text{char}: \text{prime } \text{CHAR}('a)$ **and** $q: q = \text{CHAR}('a) \wedge n$

and $\text{eq}: \text{frobenius-power } q\ x = \text{frobenius-power } q\ y$

shows $x = y$

$\langle \text{proof} \rangle$

context *Subfield*

begin

interpretation *carrier*: $\text{Field } K\ (+)\ (*)\ 0\ 1$

$\langle \text{proof} \rangle$

lemma *frobenius-power-closed*:

$x \in K \implies \text{frobenius-power } q\ x \in K$

$\langle \text{proof} \rangle$

lemma *finite-frobenius-bij-betw*:

assumes $\text{fin}: \text{finite } K$ **and** $\text{char}: \text{prime } \text{CHAR}('a)$ **and** $q: q = \text{CHAR}('a) \wedge n$

shows $\text{bij-betw } (\text{frobenius-power } q)\ K\ K$

$\langle \text{proof} \rangle$


```

lemma finite-field-frobenius-identity:
  assumes fin: finite K and xK:  $x \in K$ 
  shows frobenius-power (card K)  $x = x$ 
  <proof>

end

```

```

end

```

61 The lattice of ideals

```

theory Ideal-Lattice
  imports Ideal-Theory Order-Theory
begin

```

The ideals of a commutative ring R , ordered by inclusion, form a complete lattice: the infimum of a family of ideals is their intersection (with R , so the empty family has infimum R), and the supremum is the ideal generated by their union. Together with the subgroup lattice this exercises the abstract order theory on the two archetypal substructure lattices. As a by-product we obtain the ideal *generated by a set*, the analogue of the subgroup *generate*.

```

context commutative-ring
begin

```

61.1 Intersections of ideals

The intersection of a nonempty family of ideals is an ideal.

```

lemma ideal-Inter:
  assumes ne:  $\mathcal{A} \neq \{\}$  and id:  $\bigwedge I. I \in \mathcal{A} \implies \text{Ideal } I \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ 
  shows Ideal  $(\bigcap \mathcal{A}) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ 
  <proof>

```

The whole ring is an ideal of itself (the top of the ideal lattice).

```

lemma whole-ideal: Ideal  $R \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ 
<proof>

```

61.2 The ideal generated by a set

The ideal generated by a subset S of the ring: the intersection of all ideals containing $S \cap R$. (Intersecting with R in the guard mirrors the subgroup *generate*, so elements outside the carrier are ignored.)

```

definition ideal-generated :: 'a set  $\Rightarrow$  'a set
  where ideal-generated  $S = \bigcap \{I. \text{Ideal } I \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \wedge S \cap R \subseteq I\}$ 

```

```

lemma ideal-generated-is-ideal: Ideal  $(\text{ideal-generated } S) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ 

```

<proof>

lemma *ideal-generated-incl*: $S \cap R \subseteq \text{ideal-generated } S$
<proof>

lemma *ideal-generated-minimal*:
assumes *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \ S \cap R \subseteq J$
shows *ideal-generated* $S \subseteq J$
<proof>

lemma *ideal-generated-subset-carrier*: *ideal-generated* $S \subseteq R$
<proof>

61.3 The product of two ideals

The product $I \odot J$ of two ideals is the ideal generated by all products $i \cdot j$ with $i \in I$ and $j \in J$. Together with the ideal sum and the intersection this completes the semiring of ideals; the product is contained in the intersection (hence, unlike sum and intersection, is in general *not* a lattice operation).

definition *ideal-prod* :: 'a set $\Rightarrow$ 'a set $\Rightarrow$ 'a set (**infixl** $\odot$ 70)
where $I \odot J = \text{ideal-generated } \{i \cdot j \mid i \in I \wedge j \in J\}$

lemma *ideal-prod-is-ideal*: *Ideal* $(I \odot J) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
<proof>

lemma *ideal-prod-subset-carrier*: $I \odot J \subseteq R$
<proof>

Generators lie in the product.

lemma *ideal-prod-memI*:
assumes I : *Ideal* $I \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ **and** J : *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
and i : $i \in I$ **and** j : $j \in J$
shows $i \cdot j \in I \odot J$
<proof>

The product is contained in both factors, hence in their intersection.

lemma *ideal-prod-subset*:
assumes I : *Ideal* $I \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ **and** J : *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
shows $I \odot J \subseteq I \cap J$
<proof>

Commutativity: the generating sets coincide because the ring is commutative.

lemma *ideal-prod-commute*:
assumes I : *Ideal* $I \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ **and** J : *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
shows $I \odot J = J \odot I$
<proof>

61.4 The complete lattice of ideals

definition *ideals* :: 'a set set

where *ideals* = $\{I. \text{Ideal } I \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}\}$

lemma *ideals-iff* [simp]: $I \in \text{ideals} \longleftrightarrow \text{Ideal } I \text{ } R \text{ } (+) \text{ } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1}$
 $\langle \text{proof} \rangle$

interpretation *ideal-order*: Partial-Order *ideals* ($\subseteq$)
 $\langle \text{proof} \rangle$

interpretation *ideal-lattice*: Complete-Lattice *ideals* ($\subseteq$)
 $\langle \text{proof} \rangle$

end

end

62 Extension and contraction of ideals along a ring homomorphism

theory *Ideal-Extension*

imports *Ideal-Lattice*

begin

Along a homomorphism η of commutative rings $R \rightarrow R'$ there are two natural operations on ideals:

- *contraction* $J \mapsto \eta^{-1} J \cap R$, the preimage of an ideal of the target, which is always an ideal of the source;
- *extension* $I \mapsto \langle \eta^{-1} I \rangle$, the ideal of the target *generated by* the image of an ideal of the source. Generation is needed: the direct image $\eta^{-1} I$ need not be an ideal of R' — it is closed under addition and under multiplication by elements of $\eta^{-1} R$, but not in general by arbitrary elements of R' . (For the inclusion of the integers in the rationals, the image of the even integers is not an ideal of the rationals.)

These form an *adjoint pair* between the ideal lattices, ordered by inclusion: $\langle \eta^{-1} I \rangle \subseteq J \longleftrightarrow I \subseteq \eta^{-1} J \cap R$. So this is a genuine instance of the *Galois-Connection* locale of *Order-Theory*, and all of its consequences — the unit and counit inequalities, monotonicity of both adjoints, the semi-inverse identities, and idempotence of the induced closure and kernel operators — come for free.

The two rings are commutative because *ideal-generated* and *ideals* are defined in the *commutative-ring* context. As with the module homomorphism locale, the target ring's operations are written with primes and the source's unprimed;

the two rings are *separate* here (unlike the shared scalar ring of a module homomorphism), so no parameter is shared.

```
locale comm-ring-homomorphism = ring-homomorphism +
  source: commutative-ring R (+) (·) 0 1 +
  target: commutative-ring R' (+') (·') 0' 1'
begin
```

62.1 Contraction: the preimage of an ideal

The contraction of a subset of the target: its preimage, cut down to the source carrier.

```
definition contract :: 'b set  $\Rightarrow$  'a set
where contract J =  $\eta - ' J \cap R$ 
```

```
lemma contract-subset: contract J  $\subseteq$  R
  <proof>
```

```
lemma contract-memI:  $\llbracket a \in R; \eta a \in J \rrbracket \Longrightarrow a \in \text{contract } J$ 
  <proof>
```

```
lemma contract-memD:  $a \in \text{contract } J \Longrightarrow a \in R \wedge \eta a \in J$ 
  <proof>
```

```
lemma contract-mono:  $J \subseteq J' \Longrightarrow \text{contract } J \subseteq \text{contract } J'$ 
  <proof>
```

The preimage of an ideal is an ideal: it contains the zero because η preserves it, is closed under addition and negation because η is additive, and absorbs multiplication because η is multiplicative and J absorbs multiplication in the target.

```
theorem contract-ideal:
  assumes J: Ideal J R' (+') (·') 0' 1'
  shows Ideal (contract J) R (+) (·) 0 1
  <proof>
```

The contraction of the target's zero ideal is the kernel; contracting the whole target gives the whole source.

```
lemma contract-whole: contract R' = R
  <proof>
```

62.2 Extension: the ideal generated by the image

The extension of a subset of the source: the ideal of the target generated by its image.

```
definition extend :: 'a set  $\Rightarrow$  'b set
where extend I = target.ideal-generated ( $\eta - ' I$ )
```

lemma *extend-ideal*: *Ideal* (*extend* I) R' $(+')$ $(\cdot')$ $\mathbf{0}' \mathbf{1}'$
 $\langle \text{proof} \rangle$

lemma *extend-subset*: *extend* $I \subseteq R'$
 $\langle \text{proof} \rangle$

The image lands inside the extension. Note the intersection with R' in *target.ideal-generated-incl* is harmless: the image of a subset of R already lies in R' .

lemma *extend-incl*:
assumes $I: I \subseteq R$ **shows** $\eta \text{ ` } I \subseteq \text{extend } I$
 $\langle \text{proof} \rangle$

Extension is minimal: any target ideal containing the image contains the extension.

lemma *extend-minimal*:
assumes J : *Ideal* $J R'$ $(+')$ $(\cdot')$ $\mathbf{0}' \mathbf{1}'$ **and** *sub*: $\eta \text{ ` } I \subseteq J$
shows *extend* $I \subseteq J$
 $\langle \text{proof} \rangle$

lemma *extend-mono*:
assumes II' : $I \subseteq I'$ **and** I' : $I' \subseteq R$ **shows** *extend* $I \subseteq \text{extend } I'$
 $\langle \text{proof} \rangle$

62.3 The adjunction

Extension is left adjoint to contraction. Both directions are immediate from minimality of the generated ideal and the definition of preimage — which is exactly why extension had to be defined by generation: without it the left-to-right direction would fail.

theorem *extend-contract-galois*:
assumes $I: I \subseteq R$ **and** J : *Ideal* $J R'$ $(+')$ $(\cdot')$ $\mathbf{0}' \mathbf{1}'$
shows *extend* $I \subseteq J \longleftrightarrow I \subseteq \text{contract } J$
 $\langle \text{proof} \rangle$

The Galois connection. On the ideal lattices of the two rings, ordered by inclusion, extension and contraction form an adjoint pair. Interpreting *Galois-Connection* makes its whole development available: *unit* ($I \subseteq \text{contract } (\text{extend } I)$), *counit* ($\text{extend } (\text{contract } J) \subseteq J$), isotonicity of both maps, the semi-inverse laws $\text{extend } (\text{contract } (\text{extend } I)) = \text{extend } I$ and $\text{contract } (\text{extend } (\text{contract } J)) = \text{contract } J$, and idempotence of the closure $\text{contract} \circ \text{extend}$ and kernel $\text{extend} \circ \text{contract}$.

theorem *ideal-galois-connection*:
Galois-Connection *source.ideals* $(\subseteq)$ *target.ideals* $(\subseteq)$ *extend contract*
 $\langle \text{proof} \rangle$

The consequences of the adjunction, specialised to ideals. These are obtained by interpreting *ideal-galois-connection* locally rather than by a **sublocale** declara-

tion: a *sublocale galois-connection* inside this locale sends the interpretation machinery into a loop, because *comm-ring-homomorphism* would then be a source of *galois-connection* facts about its own ideal lattices while still being elaborated. The theorem form has no such problem, and consumers can **interpret** it wherever they need the full development.

context
begin

interpretation *GC: Galois-Connection*
 $source.ideals (\subseteq) target.ideals (\subseteq) extend contract$
 $\langle proof \rangle$

Unit: an ideal is contained in the contraction of its extension.

corollary *contract-extend-incl:*
 $Ideal\ I\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1} \implies I \subseteq contract\ (extend\ I)$
 $\langle proof \rangle$

Counit: the extension of a contraction is contained in the original ideal. The inclusion is in general strict — for $\mathbb{Z} \subseteq \mathbb{Q}$ every nonzero ideal of $\mathbb{Q}$ contracts to a nonzero ideal of $\mathbb{Z}$ whose extension is all of $\mathbb{Q}$.

corollary *extend-contract-subset:*
 $Ideal\ J\ R'\ (+')\ (\cdot')\ \mathbf{0}'\ \mathbf{1}' \implies extend\ (contract\ J) \subseteq J$
 $\langle proof \rangle$

Both adjoints are isotone on ideals.

corollary *extend-mono-ideal:*
 $\llbracket Ideal\ I\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}; Ideal\ I'\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}; I \subseteq I' \rrbracket \implies extend\ I \subseteq extend\ I'$
 $\langle proof \rangle$

corollary *contract-mono-ideal:*
 $\llbracket Ideal\ J\ R'\ (+')\ (\cdot')\ \mathbf{0}'\ \mathbf{1}'; Ideal\ J'\ R'\ (+')\ (\cdot')\ \mathbf{0}'\ \mathbf{1}'; J \subseteq J' \rrbracket$
 $\implies contract\ J \subseteq contract\ J'$
 $\langle proof \rangle$

The *semi-inverse* laws: extension and contraction are mutually inverse up to one round trip.

corollary *extend-contract-extend:*
 $Ideal\ I\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1} \implies extend\ (contract\ (extend\ I)) = extend\ I$
 $\langle proof \rangle$

corollary *contract-extend-contract:*
 $Ideal\ J\ R'\ (+')\ (\cdot')\ \mathbf{0}'\ \mathbf{1}' \implies contract\ (extend\ (contract\ J)) = contract\ J$
 $\langle proof \rangle$

Hence $contract \circ extend$ is a closure operator on the ideals of the source, and $extend \circ contract$ a kernel operator on those of the target.

corollary *closure-idempotent:*

Ideal $I\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1} \implies \text{contract} (\text{extend} (\text{contract} (\text{extend } I))) = \text{contract} (\text{extend } I)$
 ⟨proof⟩

corollary *kernel-idempotent:*

Ideal $J\ R'\ (+')\ (\cdot')\ \mathbf{0}'\ \mathbf{1}' \implies \text{extend} (\text{contract} (\text{extend} (\text{contract } J))) = \text{extend} (\text{contract } J)$
 ⟨proof⟩

end

end

end

63 The direct product of an indexed family of rings

theory *Ring-Family-Product*

imports *Ring-Theory*

begin

The external direct product of a family of rings $R\ i\ (i \in I)$: the carrier is the extensional dependent function space $Pi_E\ I\ R$, with pointwise addition, multiplication, zero and one. This mirrors the indexed group product but for rings; its commutative and field-of-components refinements follow. (A product of fields is *not* a field in general — it has zero divisors as soon as I has more than one element — so only the commutative-ring refinement is offered.)

63.1 The product locale

locale *ring-family* =

fixes $I :: 'i\ set$

and $R :: 'i \Rightarrow 'a\ set$

and $add :: 'i \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$

and $mult :: 'i \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$

and $zero :: 'i \Rightarrow 'a$

and $one :: 'i \Rightarrow 'a$

assumes *ring*: $\bigwedge i. i \in I \implies \text{Ring } (R\ i)\ (add\ i)\ (mult\ i)\ (zero\ i)\ (one\ i)$

begin

lemma *comp-ring*: $i \in I \implies \text{Ring } (R\ i)\ (add\ i)\ (mult\ i)\ (zero\ i)\ (one\ i)$

⟨proof⟩

63.2 Carrier and operations

definition *Pcarrier* :: $('i \Rightarrow 'a)\ set$

where $Pcarrier = (\Pi_E\ i \in I. R\ i)$

definition $Padd :: ('i \Rightarrow 'a) \Rightarrow ('i \Rightarrow 'a) \Rightarrow ('i \Rightarrow 'a)$ (**infixl** $\langle \oplus_{\Pi} \rangle$ 65)
where $x \oplus_{\Pi} y = (\lambda i \in I. \text{add } i \ (x \ i) \ (y \ i))$

definition $Pmult :: ('i \Rightarrow 'a) \Rightarrow ('i \Rightarrow 'a) \Rightarrow ('i \Rightarrow 'a)$ (**infixl** $\langle \odot_{\Pi} \rangle$ 70)
where $x \odot_{\Pi} y = (\lambda i \in I. \text{mult } i \ (x \ i) \ (y \ i))$

definition $Pzero :: 'i \Rightarrow 'a$ ($\langle \mathbf{0}_{\Pi} \rangle$)
where $\mathbf{0}_{\Pi} = (\lambda i \in I. \text{zero } i)$

definition $Pone :: 'i \Rightarrow 'a$ ($\langle \mathbf{1}_{\Pi} \rangle$)
where $\mathbf{1}_{\Pi} = (\lambda i \in I. \text{one } i)$

lemma $Pcarrier\text{-}memI$:
 $\llbracket \bigwedge i. i \in I \implies x \ i \in R \ i; x \in \text{extensional } I \rrbracket \implies x \in Pcarrier$
 $\langle \text{proof} \rangle$

lemma $Pcarrier\text{-}component$: $\llbracket x \in Pcarrier; i \in I \rrbracket \implies x \ i \in R \ i$
 $\langle \text{proof} \rangle$

lemma $Pcarrier\text{-}extensional$: $x \in Pcarrier \implies x \in \text{extensional } I$
 $\langle \text{proof} \rangle$

lemma $Pcarrier\text{-}eqI$:
 $\llbracket x \in Pcarrier; y \in Pcarrier; \bigwedge i. i \in I \implies x \ i = y \ i \rrbracket \implies x = y$
 $\langle \text{proof} \rangle$

lemma $Padd\text{-}apply$: $i \in I \implies (x \oplus_{\Pi} y) \ i = \text{add } i \ (x \ i) \ (y \ i)$ $\langle \text{proof} \rangle$

lemma $Pmult\text{-}apply$: $i \in I \implies (x \odot_{\Pi} y) \ i = \text{mult } i \ (x \ i) \ (y \ i)$ $\langle \text{proof} \rangle$

lemma $Pzero\text{-}apply$: $i \in I \implies \mathbf{0}_{\Pi} \ i = \text{zero } i$ $\langle \text{proof} \rangle$

lemma $Pone\text{-}apply$: $i \in I \implies \mathbf{1}_{\Pi} \ i = \text{one } i$ $\langle \text{proof} \rangle$

lemma $Padd\text{-}closed$ [*intro*, *simp*]:
assumes $x \in Pcarrier \ y \in Pcarrier$ **shows** $x \oplus_{\Pi} y \in Pcarrier$
 $\langle \text{proof} \rangle$

lemma $Pmult\text{-}closed$ [*intro*, *simp*]:
assumes $x \in Pcarrier \ y \in Pcarrier$ **shows** $x \odot_{\Pi} y \in Pcarrier$
 $\langle \text{proof} \rangle$

lemma $Pzero\text{-}closed$ [*intro*, *simp*]: $\mathbf{0}_{\Pi} \in Pcarrier$
 $\langle \text{proof} \rangle$

lemma $Pone\text{-}closed$ [*intro*, *simp*]: $\mathbf{1}_{\Pi} \in Pcarrier$
 $\langle \text{proof} \rangle$

63.3 The additive abelian group

lemma $additive\text{-}group$: $\text{Group } Pcarrier \ (\oplus_{\Pi}) \ \mathbf{0}_{\Pi}$
 $\langle \text{proof} \rangle$

lemma *additive-abelian-group: Abelian-Group Pcarrier* $(\oplus_{\Pi}) \mathbf{0}_{\Pi}$
 $\langle proof \rangle$

63.4 The multiplicative monoid

lemma *multiplicative-monoid: Monoid Pcarrier* $(\odot_{\Pi}) \mathbf{1}_{\Pi}$
 $\langle proof \rangle$

63.5 The product is a ring

theorem *Ring-Pcarrier: Ring Pcarrier* $(\oplus_{\Pi}) (\odot_{\Pi}) \mathbf{0}_{\Pi} \mathbf{1}_{\Pi}$
 $\langle proof \rangle$

sublocale *product: Ring Pcarrier* $(\oplus_{\Pi}) (\odot_{\Pi}) Pzero Pone$
 $\langle proof \rangle$

63.6 Projections are ring homomorphisms

definition $Pproj :: 'i \Rightarrow ('i \Rightarrow 'a) \Rightarrow 'a$
where $Pproj\ i = restrict\ (\lambda x. x\ i)\ Pcarrier$

lemma *Pproj-apply: $x \in Pcarrier \implies Pproj\ i\ x = x\ i$*
 $\langle proof \rangle$

lemma *Pproj-hom:*
assumes $i: i \in I$
shows *ring-homomorphism* $(Pproj\ i)\ Pcarrier\ (\oplus_{\Pi}) (\odot_{\Pi}) \mathbf{0}_{\Pi} \mathbf{1}_{\Pi}$
 $(R\ i)\ (add\ i)\ (mult\ i)\ (zero\ i)\ (one\ i)$
 $\langle proof \rangle$

end

63.7 The product of a family of commutative rings is commutative

locale *commutative-ring-family = ring-family +*
assumes $comm: \bigwedge i. i \in I \implies commutative-ring\ (R\ i)\ (add\ i)\ (mult\ i)\ (zero\ i)\ (one\ i)$
begin

theorem *commutative-ring-Pcarrier: commutative-ring Pcarrier* $(\oplus_{\Pi}) (\odot_{\Pi}) \mathbf{0}_{\Pi} \mathbf{1}_{\Pi}$
 $\langle proof \rangle$

end

end

64 The Chinese Remainder Theorem for commutative rings

```
theory Chinese-Remainder-Rings
imports Ideal-Theory Ring-Family-Product
begin
```

For two comaximal ideals I, J of a commutative ring R (i.e. $I + J = R$), the quotient $R / (I \cap J)$ is isomorphic to the product ring $(R / I) \times (R / J)$. The proof follows the classical route: the canonical map $a \mapsto (I + a, J + a)$ into the product of the two quotient rings is a ring homomorphism, surjective by comaximality, with kernel $I \cap J$; the fundamental theorem of ring homomorphisms then yields the isomorphism.

64.1 Associative-commutative rewriting in an abelian group

The *Abelian-Group* locale exposes only the raw associativity, commutativity and inverse axioms (each with a carrier side condition), so plain *simp* cannot ac-normalise sums. We prove a few confluent cancellation lemmas and a couple of rearrangement lemmas that the Chinese-Remainder arithmetic below relies on.

```
context Abelian-Group
begin
```

Left-nested inverse cancellations (safe as *simp* rules: not permutative).

```
lemma add-assoc-inverse-left [simp]:
   $\llbracket a \in G; b \in G \rrbracket \implies \text{inverse } a \cdot (a \cdot b) = b$ 
   $\langle \text{proof} \rangle$ 
```

```
lemma add-assoc-inverse-left' [simp]:
   $\llbracket a \in G; b \in G \rrbracket \implies a \cdot (\text{inverse } a \cdot b) = b$ 
   $\langle \text{proof} \rangle$ 
```

Rearrangement: $q \cdot p \cdot \text{local.inverse } (s \cdot q) = p \cdot \text{local.inverse } s$. This is the shape needed to simplify $(x \cdot j + y \cdot i) - (x \cdot i + x \cdot j)$ in the surjectivity argument.

```
lemma add-diff-cancel-mid:
  assumes  $q: q \in G$  and  $p: p \in G$  and  $s: s \in G$ 
  shows  $(q \cdot p) \cdot \text{inverse } (s \cdot q) = p \cdot \text{inverse } s$ 
   $\langle \text{proof} \rangle$ 
```

```
end
```

64.2 The binary direct product of two rings

The external direct product of two rings, carried on pairs, with pointwise operations. This specialises the indexed direct product of *Ring-Family-Product*

to two factors, which keeps the carrier type $'a \times 'b$ rather than a dependent function space — convenient for the two quotient rings occurring in the Chinese Remainder Theorem. The notation follows *ring-homomorphism*: primed operations for the second factor.

```

locale ring-pair =
  A: Ring R (+) (·) 0 1 + B: Ring R' (+') (·') 0' 1'
  for R and addition (infixl <+> 65) and multiplication (infixl <·> 70) and zero
  (<0>) and unit (<1>)
  and R' and addition' (infixl <+'> 65) and multiplication' (infixl <·'> 70) and
  zero' (<0''>) and unit' (<1''>)
begin

```

```

definition Bcarrier :: ('a × 'b) set
  where Bcarrier = R × R'

```

```

definition Badd :: ('a × 'b) ⇒ ('a × 'b) ⇒ ('a × 'b) (infixl <⊕×> 65)
  where x ⊕× y = (fst x + fst y, snd x + ' snd y)

```

```

definition Bmult :: ('a × 'b) ⇒ ('a × 'b) ⇒ ('a × 'b) (infixl <⊗×> 70)
  where x ⊗× y = (fst x · fst y, snd x · ' snd y)

```

```

definition Bzero :: 'a × 'b (<0×>)
  where 0× = (0, 0')

```

```

definition Bone :: 'a × 'b (<1×>)
  where 1× = (1, 1')

```

```

lemma Bcarrier-iff [simp]: (a, b) ∈ Bcarrier ⟷ a ∈ R ∧ b ∈ R'
  <proof>

```

```

lemma Badd-apply [simp]: (a1, b1) ⊕× (a2, b2) = (a1 + a2, b1 + ' b2)
  <proof>

```

```

lemma Bmult-apply [simp]: (a1, b1) ⊗× (a2, b2) = (a1 · a2, b1 · ' b2)
  <proof>

```

```

theorem Ring-Bcarrier: Ring Bcarrier (⊕×) (⊗×) 0× 1×
  <proof>

```

```

sublocale product: Ring Bcarrier (⊕×) (⊗×) Bzero Bone
  <proof>

```

```

end

```

64.3 Sum of two ideals

The sum $I + J$ of two ideals: all sums $i + j$ with $i \in I, j \in J$. We work in a commutative ring throughout (the setting of the Chinese Remainder Theorem).

context *ideal-in-comm-ring*

begin

definition *ideal-sum* :: 'a set $\Rightarrow$ 'a set

where *ideal-sum* $J = \{i + j \mid i \in I \wedge j \in J\}$

lemma *ideal-sum-memI*: $\llbracket i \in I; j \in J \rrbracket \Longrightarrow i + j \in \text{ideal-sum } J$
 $\langle \text{proof} \rangle$

lemma *ideal-sum-memE*:

assumes $x \in \text{ideal-sum } J$

obtains $i \ j$ **where** $i \in I \ j \in J \ x = i + j$

$\langle \text{proof} \rangle$

The sum contains both summands (using that ideals contain $\mathbf{0}$).

lemma *I-subset-ideal-sum*:

assumes *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ **shows** $I \subseteq \text{ideal-sum } J$
 $\langle \text{proof} \rangle$

lemma *J-subset-ideal-sum*:

assumes *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ **shows** $J \subseteq \text{ideal-sum } J$
 $\langle \text{proof} \rangle$

The intersection of the ambient ideal I with another ideal is again an ideal.

lemma *inter-ideal*:

assumes *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

shows *Ideal* $(I \cap J) \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

$\langle \text{proof} \rangle$

Comaximality is preserved under intersection: if I is comaximal with J and with K (meaning $\mathbf{1} \in \text{ideal-sum } J$ and $\mathbf{1} \in \text{ideal-sum } K$), then I is comaximal with $J \cap K$. The key step multiplies the two witnesses $i_1 + j = \mathbf{1}$, $i_2 + k = \mathbf{1}$ and regroups.

lemma *comaximal-inter*:

assumes J : *Ideal* $J \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$ **and** K : *Ideal* $K \ R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

and *comI* J : $\exists i \in I. \exists j \in J. i + j = \mathbf{1}$

and *comI* K : $\exists i \in I. \exists k \in K. i + k = \mathbf{1}$

shows $\exists i \in I. \exists m \in J \cap K. i + m = \mathbf{1}$

$\langle \text{proof} \rangle$

The reduction step underlying Chinese-Remainder surjectivity, phrased for a single ideal (where subtraction notation is unambiguous): if $i + j = \mathbf{1}$ with $i \in I$, then the element $x \cdot j + y \cdot i$ has the same coset as x modulo I (because $x \cdot j \equiv x$ and $y \cdot i \equiv \mathbf{0}$ modulo I).

lemma *coset-comaximal*:

assumes x : $x \in R$ **and** y : $y \in R$ **and** i : $i \in I$ **and** j : $j \in R$ **and** ij : $i + j = \mathbf{1}$

shows *coset* $(x \cdot j + y \cdot i) = \text{coset } x$

$\langle \text{proof} \rangle$

end

64.4 Two comaximal ideals

The Chinese Remainder Theorem setting: a commutative ring R with two ideals I, J that are *comaximal* ($I + J = R$). We package these in a locale, giving each ideal its own quotient-ring interface ($I.$ and $J.$).

```

locale comaximal-ideals =
  I: quotient-ring I R (+) (·) 0 1 + J: quotient-ring J R (+) (·) 0 1
  for I and J and R and addition (infixl <+> 65) and multiplication (infixl <·> 70)
  and zero (<0>) and unit (<1>) +
  assumes comm:  $\llbracket a \in R; b \in R \rrbracket \implies a \cdot b = b \cdot a$ 
  and comaximal:  $\exists i \in I. \exists j \in J. i + j = 1$ 
begin

```

Both ideals live in a commutative ring; expose the commutative-ring structure and the *ideal-in-comm-ring* facts for each.

```

sublocale I: ideal-in-comm-ring I R (+) (·) 0 1
  <proof>

```

```

sublocale J: ideal-in-comm-ring J R (+) (·) 0 1
  <proof>

```

The product ring $(R/I) \times (R/J)$ of the two quotient rings.

```

sublocale prod: ring-pair
  R / I.Ring-Congruence I.additive.quotient-composition I.multiplicative.quotient-composition
  I.additive.Class 0 I.additive.Class 1
  R / J.Ring-Congruence J.additive.quotient-composition J.multiplicative.quotient-composition
  J.additive.Class 0 J.additive.Class 1
  <proof>

```

The canonical projection $a \mapsto (I + a, J + a)$ into the product of the two quotients.

```

definition canon :: 'a  $\Rightarrow$  'a set  $\times$  'a set
  where canon = restrict ( $\lambda a. (I.\text{coset } a, J.\text{coset } a)$ ) R

```

```

lemma canon-apply [simp]:  $a \in R \implies \text{canon } a = (I.\text{coset } a, J.\text{coset } a)$ 
  <proof>

```

```

lemma coset-I-in-carrier:  $a \in R \implies I.\text{coset } a \in R / I.\text{Ring-Congruence}$ 
  <proof>

```

```

lemma coset-J-in-carrier:  $a \in R \implies J.\text{coset } a \in R / J.\text{Ring-Congruence}$ 
  <proof>

```

The canonical projection is a ring homomorphism into the product ring.

```

theorem canon-hom:

```

*ring-homomorphism canon R (+) (·) 0 1 prod.Bcarrier (prod.Badd) (prod.Bmult)
 prod.Bzero prod.Bone
 <proof>*

interpretation *canon: ring-homomorphism canon R (+) (·) 0 1
 prod.Bcarrier prod.Badd prod.Bmult prod.Bzero prod.Bone
 <proof>*

The kernel of the canonical projection is $I \cap J$: an element maps to zero exactly when both its cosets vanish, i.e. it lies in both ideals.

theorem *canon-kernel: canon.additive.Ker = $I \cap J$
 <proof>*

Comaximality makes the canonical projection surjective: given target cosets of x and y , the element $x \cdot j + y \cdot i$ (where $i + j = \mathbf{1}$, $i \in I$, $j \in J$) reduces to $x \bmod I$ and to $y \bmod J$.

theorem *canon-surjective: canon ‘ $R = \text{prod.Bcarrier}$
 <proof>*

64.5 The Chinese Remainder Theorem

Feeding the canonical projection to the fundamental theorem of ring homomorphisms yields R / Ker canon isomorphic to the image, which surjectivity identifies with the whole product ring, and the kernel is $I \cap J$.

interpretation *canon: ring-homomorphism-fundamental canon R (+) (·) 0 1
 prod.Bcarrier prod.Badd prod.Bmult prod.Bzero prod.Bone
 <proof>*

The induced map $\eta|$ is an isomorphism from R / Ker onto the image; surjectivity identifies the image with the whole product ring. With *canon-kernel* this is $R / (I \cap J) \cong (R / I) \times (R / J)$.

theorem *chinese-remainder:
 ($R / (\text{subgroup-of-additive-group-of-ring.Ring-Congruence } (I \cap J) R (+) \mathbf{0})$,
 canon.kernel.additive.quotient-composition, canon.kernel.multiplicative.quotient-composition,
 canon.kernel.additive.Class $\mathbf{0}$, canon.kernel.additive.Class $\mathbf{1}$)
 $\cong_R (\text{prod.Bcarrier}, \text{prod.Badd}, \text{prod.Bmult}, \text{prod.Bzero}, \text{prod.Bone})$
 <proof>*

end

Transfer of the ring structure along operations that agree on the carrier. If R is a ring under $(+)$, $(\cdot)$ and the alternative operations $(\oplus)$, $(\otimes)$ agree with them on all pairs of carrier elements, then R is also a ring under $(\oplus)$, $(\otimes)$.

lemma *Ring-cong-on-carrier:
 fixes R and addition (infixl <+> 65) and multiplication (infixl <·> 70)
 and addition' (infixl <⊕> 65) and multiplication' (infixl <⊗> 70) and zero (<0>)
 and unit (<1>)*

```

assumes ring: Ring  $R$   $(+)$   $(\cdot)$  0 1
  and add-eq:  $\bigwedge a\ b. \llbracket a \in R; b \in R \rrbracket \implies a \oplus b = a + b$ 
  and mult-eq:  $\bigwedge a\ b. \llbracket a \in R; b \in R \rrbracket \implies a \otimes b = a \cdot b$ 
shows Ring  $R$   $(\oplus)$   $(\otimes)$  0 1
<proof>

```

65 The n-ary Chinese Remainder Theorem

Generalise the binary CRT to a finite family of pairwise-comaximal ideals: $R / (\bigcap_{i \in S} J\ i) \cong_R \prod_{i \in S} R / J\ i$. The proof proceeds by induction on the finite index set, reducing $(n+1)$ ideals to the binary theorem via the lemma that comaximality is preserved under intersection.

65.1 Pairwise-comaximal ideals

```

locale pairwise-comaximal-ideals =
  fixes  $S :: 'i\ set$ 
    and  $J :: 'i \Rightarrow 'a\ set$ 
    and  $R :: 'a\ set$ 
    and addition (infixl  $\langle + \rangle$  65)
    and multiplication (infixl  $\langle \cdot \rangle$  70)
    and zero ( $\langle 0 \rangle$ )
    and unit ( $\langle 1 \rangle$ )
assumes fin: finite  $S$ 
    and nonempty:  $S \neq \{\}$ 
    and comm:  $\llbracket a \in R; b \in R \rrbracket \implies a \cdot b = b \cdot a$ 
    and ideals:  $\bigwedge i. i \in S \implies \text{quotient-ring } (J\ i)\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$ 
    and pairwise:  $\bigwedge i\ j. \llbracket i \in S; j \in S; i \neq j \rrbracket \implies \exists a \in J\ i. \exists b \in J\ j. a + b = \mathbf{1}$ 
begin

sublocale Ring  $R$   $(+)$   $(\cdot)$  0 1
<proof>

```

```

sublocale commutative-ring  $R$   $(+)$   $(\cdot)$  0 1
<proof>

```

Each $J\ i$ is an ideal in the commutative ring.

```

lemma Ji-ideal:  $i \in S \implies \text{Ideal } (J\ i)\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$ 
<proof>

```

```

lemma Ji-ideal-comm:  $i \in S \implies \text{ideal-in-comm-ring } (J\ i)\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$ 
<proof>

```

The intersection $\bigcap (J\ ` T)$ over a subset $T \subseteq S$ is again an ideal.

```

lemma Inter-ideal:
  assumes TS:  $T \subseteq S$  and The:  $T \neq \{\}$ 
  shows Ideal  $(\bigcap (J\ ` T))\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$ 

```

<proof>

Key induction step: if $J\ i$ is comaximal with every $J\ j$ for $j \in T$, then $J\ i$ is comaximal with $\bigcap (J\ 'T)$.

lemma *comaximal-with-Inter:*

assumes $iS: i \in S$ **and** $TS: T \subseteq S$ **and** $One: T \neq \{\}$ **and** $iT: i \notin T$

and $pw: \bigwedge j. j \in T \implies \exists a \in J\ i. \exists b \in J\ j. a + b = 1$

shows $\exists a \in J\ i. \exists b \in \bigcap (J\ 'T). a + b = 1$

<proof>

65.2 The canonical projection into the indexed product

The quotient rings form a ring family. The coset *Ji-Class* $i\ a$ is the equivalence class of a modulo the ring congruence induced by $J\ i$. The quotient carrier, operations, and constants are defined in terms of this class map.

definition *Ji-Cong* $:: 'i \Rightarrow ('a \times 'a)$ **set where**

$Ji-Cong\ i = \{(a, b). a \in R \wedge b \in R \wedge a - b \in J\ i\}$

definition *Ji-Class* $:: 'i \Rightarrow 'a \Rightarrow 'a$ **set where**

$Ji-Class\ i\ a = \{b \in R. (a, b) \in Ji-Cong\ i\}$

definition *quot-carrier* $:: 'i \Rightarrow 'a$ **set set where**

$quot-carrier\ i = Ji-Class\ i\ 'R$

definition *quot-add* $:: 'i \Rightarrow 'a\ set \Rightarrow 'a\ set \Rightarrow 'a\ set$ **where**

$quot-add\ i\ A\ B = (THE\ C. \exists a \in A. \exists b \in B. C = Ji-Class\ i\ (a + b))$

definition *quot-mult* $:: 'i \Rightarrow 'a\ set \Rightarrow 'a\ set \Rightarrow 'a\ set$ **where**

$quot-mult\ i\ A\ B = (THE\ C. \exists a \in A. \exists b \in B. C = Ji-Class\ i\ (a \cdot b))$

definition *quot-zero* $:: 'i \Rightarrow 'a$ **set where**

$quot-zero\ i = Ji-Class\ i\ 0$

definition *quot-one* $:: 'i \Rightarrow 'a$ **set where**

$quot-one\ i = Ji-Class\ i\ 1$

lemma *Ji-Cong-eq:*

assumes $i \in S$

shows $Ji-Cong\ i = subgroup-of-additive-group-of-ring.Ring-Congruence\ (J\ i)\ R\ addition\ 0$

<proof>

Key bridge: *Ji-Class* agrees with the quotient-ring coset. The proof is by unfolding definitions within a local interpretation.

lemma *Ji-Class-unfold:*

assumes $a \in R$

shows $Ji-Class\ i\ a = \{b \in R. a - b \in J\ i\}$

<proof>

lemma *Ji-Cong-sym*:
assumes $i: i \in S$ **and** $(a, b) \in \text{Ji-Cong } i$
shows $(b, a) \in \text{Ji-Cong } i$
 $\langle \text{proof} \rangle$

lemma *Ji-Class-mem-sym*:
assumes $i: i \in S$ **and** $a \in R$ $b \in \text{Ji-Class } i$ a
shows $a \in \text{Ji-Class } i$ b
 $\langle \text{proof} \rangle$

lemma *Ji-Class-coset-bridge*:
assumes $i: i \in S$ **and** $aR: a \in R$
shows $\text{Ji-Class } i$ $a = \{b \in R. (a, b) \in \text{Ji-Cong } i\}$
 $\langle \text{proof} \rangle$

Within an interpretation qi : *quotient-ring* $(J\ i)\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$, the coset $qi.\text{coset}$ a equals $\text{Ji-Class } i$ a . We package this as a fact to use in proofs.

lemma *Ji-Class-qi-coset*:
assumes $i: i \in S$ **and** $aR: a \in R$
shows $\text{Ji-Class } i$ $a = \{b \in R. a - b \in J\ i\}$
 $\langle \text{proof} \rangle$

The coset map is well-defined modulo the congruence.

lemma *Ji-Class-cong*:
assumes $i: i \in S$ **and** $a \in R$ $b \in R$ $(a, b) \in \text{Ji-Cong } i$
shows $\text{Ji-Class } i$ $a = \text{Ji-Class } i$ b
 $\langle \text{proof} \rangle$

lemma *Ji-Class-repr*:
assumes $i: i \in S$ **and** $A \in \text{quot-carrier } i$
obtains a **where** $a \in R$ $A = \text{Ji-Class } i$ a
 $\langle \text{proof} \rangle$

lemma *quot-Ring*:
assumes $i: i \in S$
shows $\text{Ring } (\text{quot-carrier } i) (\text{quot-add } i) (\text{quot-mult } i) (\text{quot-zero } i) (\text{quot-one } i)$
 $\langle \text{proof} \rangle$

interpretation *prod*: *ring-family* S *quot-carrier* *quot-add* *quot-mult* *quot-zero* *quot-one*
 $\langle \text{proof} \rangle$

The coset map for each component — directly uses *Ji-Class*.

definition *ncoset* :: $'i \Rightarrow 'a \Rightarrow 'a$ **set** **where**
 $\text{ncoset } i$ $a = \text{Ji-Class } i$ a

lemma *ncoset-in-carrier*:
assumes $i \in S$ $a \in R$
shows $\text{ncoset } i$ $a \in \text{quot-carrier } i$

$\langle \text{proof} \rangle$

The canonical map $a \mapsto (\lambda i \in S. \text{Ji-Class } i \ a)$ into the indexed product.

definition $\text{ncanon} :: 'a \Rightarrow ('i \Rightarrow 'a \text{ set})$ **where**
 $\text{ncanon} = \text{restrict } (\lambda a. \lambda i \in S. \text{ncoset } i \ a) \ R$

lemma ncanon-apply :
assumes $a \in R \ i \in S$
shows $\text{ncanon } a \ i = \text{ncoset } i \ a$
 $\langle \text{proof} \rangle$

lemma ncanon-closed :
assumes $a \in R$
shows $\text{ncanon } a \in \text{prod.Pcarrier}$
 $\langle \text{proof} \rangle$

lemma ncanon-outside : $a \notin R \implies \text{ncanon } a = \text{undefined}$
 $\langle \text{proof} \rangle$

The canonical map is a ring homomorphism.

theorem ncanon-hom :
 $\text{ring-homomorphism } \text{ncanon } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
 $\text{prod.Pcarrier } \text{prod.Padd } \text{prod.Pmult } \text{prod.Pzero } \text{prod.Pone}$
 $\langle \text{proof} \rangle$

interpretation ncanon : $\text{ring-homomorphism } \text{ncanon } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
 $\text{prod.Pcarrier } \text{prod.Padd } \text{prod.Pmult } \text{prod.Pzero } \text{prod.Pone}$
 $\langle \text{proof} \rangle$

interpretation ncanon : $\text{ring-homomorphism-fundamental } \text{ncanon } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
 $\text{prod.Pcarrier } \text{prod.Padd } \text{prod.Pmult } \text{prod.Pzero } \text{prod.Pone}$
 $\langle \text{proof} \rangle$

65.3 Kernel and surjectivity

The kernel of the canonical projection is $\bigcap (J \text{ ' } S)$.

theorem ncanon-kernel : $\text{ncanon.additive.Ker} = \bigcap (J \text{ ' } S)$
 $\langle \text{proof} \rangle$

The *Ji-Class* version of $\llbracket \text{ideal-in-comm-ring } ?I \ ?R \ ?\text{addition } ?\text{multiplication} \ ?\text{zero } ?\text{unit}; \ ?x \in ?R; \ ?y \in ?R; \ ?i \in ?I; \ ?j \in ?R; \ ?\text{addition } ?i \ ?j = ?\text{unit} \rrbracket \implies$
 $\text{Ideal.coset } ?I \ ?R \ ?\text{addition } ?\text{zero } (? \text{addition } (? \text{multiplication } ?x \ ?j) \ (? \text{multiplication } ?y \ ?i)) = \text{Ideal.coset } ?I \ ?R \ ?\text{addition } ?\text{zero } ?x$: if $m + n = \mathbf{1}$ with $m \in J \ i$,
then $x \cdot n + y \cdot m$ and x have the same class modulo $J \ i$.

lemma $\text{Ji-Class-comaximal}$:
assumes $i: i \in S$ **and** $x: x \in R$ **and** $y: y \in R$
and $m: m \in J \ i$ **and** $nR: n \in R$ **and** $mn: m + n = \mathbf{1}$
shows $\text{Ji-Class } i \ (x \cdot n + y \cdot m) = \text{Ji-Class } i \ x$

<proof>

Surjectivity of the canonical map. We first prove, by induction on a nonempty finite subset $T \subseteq S$, that any family of target classes over T is realised simultaneously by a single element of R . The inductive step splits off one index k and combines a solution over the rest via the comaximality of $J \ k$ with $\bigcap (J \text{ ' } F)$ ($\llbracket ?i \in S; ?T \subseteq S; ?T \neq \{\}; ?i \notin ?T; \bigwedge j. j \in ?T \implies \exists a \in J \ ?i. \exists b \in J \ j. a + b = \mathbf{1} \rrbracket \implies \exists a \in J \ ?i. \exists b \in \bigcap (J \text{ ' } ?T). a + b = \mathbf{1}$).

lemma *partial-solution*:

assumes $T \subseteq S$ **and** $T \neq \{\}$ **and** $\bigwedge i. i \in T \implies x \ i \in R$
shows $\exists a \in R. \forall i \in T. \text{Ji-Class } i \ a = \text{Ji-Class } i \ (x \ i)$

<proof>

theorem *ncanon-surjective*: $\text{ncanon} \text{ ' } R = \text{prod.Pcarrier}$

<proof>

65.4 The n-ary Chinese Remainder Theorem

theorem *chinese-remainder-general*:

$(R \ / \ (\text{subgroup-of-additive-group-of-ring.Ring-Congruence } (\bigcap (J \text{ ' } S)) \ R \ (+) \ \mathbf{0}),$
 $\text{ncanon.kernel.additive.quotient-composition}, \text{ncanon.kernel.multiplicative.quotient-composition},$
 $\text{ncanon.kernel.additive.Class } \mathbf{0}, \text{ncanon.kernel.additive.Class } \mathbf{1})$
 $\cong_R (\text{prod.Pcarrier}, \text{prod.Padd}, \text{prod.Pmult}, \text{prod.Pzero}, \text{prod.Pone})$

<proof>

end

end

66 The field of fractions of an integral domain

theory *Field-Of-Fractions*

imports *Divisibility-Theory*

begin

Given an integral domain R we construct its field of fractions as a carrier-set object, mirroring the elementary construction of $\mathbb{Q}$ from $\mathbb{Z}$: fractions are pairs (a, b) with $b \neq \mathbf{0}$, identified when they cross-multiply equal, and the field operations are the usual ones on numerators and denominators. We reuse the *Equivalence* machinery of *Set-Theory* for the quotient, then interpret *Field* on it. The canonical map $a \mapsto a / \mathbf{1}$ is an injective ring homomorphism, giving the universal embedding of the domain into a field.

context *integral-domain*

begin

66.1 Fractions and their equivalence

A fraction is a pair of ring elements with nonzero denominator.

definition *frac-set* :: ($'a \times 'a$) set
where *frac-set* = $\{(a, b). a \in R \wedge b \in R \wedge b \neq \mathbf{0}\}$

lemma *frac-setI* [intro]: $\llbracket a \in R; b \in R; b \neq \mathbf{0} \rrbracket \implies (a, b) \in \text{frac-set}$
 $\langle \text{proof} \rangle$

lemma *frac-setD* [dest]: $(a, b) \in \text{frac-set} \implies a \in R \wedge b \in R \wedge b \neq \mathbf{0}$
 $\langle \text{proof} \rangle$

The denominator of a fraction is a nonzero ring element; the product of two nonzero denominators is again nonzero (no zero divisors).

lemma *frac-mult-denom-nonzero* [intro]:
 $\llbracket b \in R; d \in R; b \neq \mathbf{0}; d \neq \mathbf{0} \rrbracket \implies b \cdot d \neq \mathbf{0}$
 $\langle \text{proof} \rangle$

Cross-multiplication equivalence: $(a, b) \sim (c, d)$ iff $a \cdot d = c \cdot b$.

definition *frac-rel* :: $((a \times 'a) \times (c \times 'a))$ set
where *frac-rel* = $\{((a, b), (c, d)). (a, b) \in \text{frac-set} \wedge (c, d) \in \text{frac-set} \wedge a \cdot d = c \cdot b\}$

lemma *frac-relI* [intro]:
 $\llbracket (a, b) \in \text{frac-set}; (c, d) \in \text{frac-set}; a \cdot d = c \cdot b \rrbracket \implies ((a, b), (c, d)) \in \text{frac-rel}$
 $\langle \text{proof} \rangle$

lemma *frac-relE*:
assumes $((a, b), (c, d)) \in \text{frac-rel}$
obtains $(a, b) \in \text{frac-set} \ (c, d) \in \text{frac-set} \ a \cdot d = c \cdot b$
 $\langle \text{proof} \rangle$

Transitivity is where cancellation — hence the domain hypothesis — is used.

lemma *frac-rel-trans*:
assumes 1: $((a, b), (c, d)) \in \text{frac-rel}$ **and** 2: $((c, d), (e, f)) \in \text{frac-rel}$
shows $((a, b), (e, f)) \in \text{frac-rel}$
 $\langle \text{proof} \rangle$

The relation is an equivalence on the set of fractions.

interpretation *frac*: *Equivalence frac-set frac-rel*
 $\langle \text{proof} \rangle$

66.2 The carrier and class equality

The carrier of the field of fractions: the set of equivalence classes.

abbreviation *Fractions* :: $('a \times 'a)$ set set
where *Fractions* $\equiv \text{frac-set} / \text{frac-rel}$

Class equality is cross-multiplication.

lemma *frac-eqI*:

assumes $(a, b) \in \text{frac-set } (c, d) \in \text{frac-set } a \cdot d = c \cdot b$
shows $\text{frac.Class } (a, b) = \text{frac.Class } (c, d)$
 $\langle \text{proof} \rangle$

lemma *frac-eqD*:

assumes $(a, b) \in \text{frac-set } (c, d) \in \text{frac-set } \text{frac.Class } (a, b) = \text{frac.Class } (c, d)$
shows $a \cdot d = c \cdot b$
 $\langle \text{proof} \rangle$

A class lies in the carrier exactly when it is the class of a fraction.

lemma *Class-in-Fractions* [intro, simp]: $(a, b) \in \text{frac-set} \implies \text{frac.Class } (a, b) \in \text{Fractions}$
 $\langle \text{proof} \rangle$

lemma *Fractions-cases*:

assumes $A \in \text{Fractions}$ **obtains** $a \ b$ **where** $(a, b) \in \text{frac-set } A = \text{frac.Class } (a, b)$
 $\langle \text{proof} \rangle$

A class equals the zero-class $\text{Class } (0, 1)$ iff any (equivalently every) numerator is zero. Later, we take $\text{Class } (0, 1)$ as the additive unit and $\text{Class } (1, 1)$ as the multiplicative unit of *Fractions*.

lemma *frac-Class-zero-iff*:

assumes $ab: (a, b) \in \text{frac-set}$
shows $\text{frac.Class } (a, b) = \text{frac.Class } (0, 1) \longleftrightarrow a = 0$
 $\langle \text{proof} \rangle$

66.3 The operations on classes

We define addition, multiplication, negation, and inversion on classes by picking any representative and applying the elementary fraction formulas. For each operation we prove a congruence lemma (well-definedness under the equivalence) and a *Class*-formula giving the computation on a canonical representative.

definition *frac-add* :: $('a \times 'a) \text{ set} \Rightarrow ('a \times 'a) \text{ set} \Rightarrow ('a \times 'a) \text{ set}$

(**infixl** $\langle [+]\rangle$ 65)

where $A \ [+]\ B =$

($\text{let } (a, b) = \text{SOME } p. p \in A \wedge p \in \text{frac-set};$
 $(c, d) = \text{SOME } p. p \in B \wedge p \in \text{frac-set}$
 $\text{in } \text{frac.Class } (a \cdot d + c \cdot b, b \cdot d)$)

definition *frac-mult* :: $('a \times 'a) \text{ set} \Rightarrow ('a \times 'a) \text{ set} \Rightarrow ('a \times 'a) \text{ set}$

(**infixl** $\langle [\cdot]\rangle$ 70)

where $A \ [\cdot]\ B =$

($\text{let } (a, b) = \text{SOME } p. p \in A \wedge p \in \text{frac-set};$
 $(c, d) = \text{SOME } p. p \in B \wedge p \in \text{frac-set}$
 $\text{in } \text{frac.Class } (a \cdot c, b \cdot d)$)

definition *frac-neg* :: $('a \times 'a) \text{ set} \Rightarrow ('a \times 'a) \text{ set} \Rightarrow ('a \times 'a) \text{ set}$

where $[-]\ A = (\text{let } (a, b) = \text{SOME } p. p \in A \wedge p \in \text{frac-set} \text{ in } \text{frac.Class } (- a, b))$

Inversion is only used on nonzero classes.

definition *frac-inv* :: ('a × 'a) set ⇒ ('a × 'a) set
where *frac-inv* A = (let (a, b) = SOME p. p ∈ A ∧ p ∈ *frac-set* in *frac.Class* (b, a))

The zero and unit of the fraction field.

abbreviation *frac-zero* :: ('a × 'a) set (⟨[0]⟩)
where [0] ≡ *frac.Class* (0, 1)

abbreviation *frac-one* :: ('a × 'a) set (⟨[1]⟩)
where [1] ≡ *frac.Class* (1, 1)

lemma *one-in-frac-set*: (1, 1) ∈ *frac-set*
 ⟨proof⟩

lemma *zero-in-frac-set*: (0, 1) ∈ *frac-set*
 ⟨proof⟩

lemma *frac-zero-in* [intro, simp]: [0] ∈ *Fractions*
 ⟨proof⟩

lemma *frac-one-in* [intro, simp]: [1] ∈ *Fractions*
 ⟨proof⟩

lemma *frac-one-neq-zero*: [1] ≠ [0]
 ⟨proof⟩

66.3.1 Representative selection

Any class of a fraction contains that fraction.

lemma *rep-in-frac-set*:
assumes (a, b) ∈ *frac-set*
shows (SOME p. p ∈ *frac.Class* (a, b) ∧ p ∈ *frac-set*) ∈ *frac.Class* (a, b) ∧
 (SOME p. p ∈ *frac.Class* (a, b) ∧ p ∈ *frac-set*) ∈ *frac-set*
 ⟨proof⟩

66.3.2 Congruence: well-definedness under the equivalence

lemma *frac-add-cong*:
assumes r1: ((a, b), (a', b')) ∈ *frac-rel* **and** r2: ((c, d), (c', d')) ∈ *frac-rel*
shows ((a · d + c · b, b · d), (a' · d' + c' · b', b' · d')) ∈ *frac-rel*
 ⟨proof⟩

lemma *frac-mult-cong*:
assumes r1: ((a, b), (a', b')) ∈ *frac-rel* **and** r2: ((c, d), (c', d')) ∈ *frac-rel*
shows ((a · c, b · d), (a' · c', b' · d')) ∈ *frac-rel*
 ⟨proof⟩

lemma *frac-neg-cong*:

assumes $r: ((a, b), (a', b')) \in \text{frac-rel}$
shows $((- a, b), (- a', b')) \in \text{frac-rel}$
 $\langle \text{proof} \rangle$

Inversion: well-defined provided the numerators are nonzero (i.e. the class is nonzero).

lemma *frac-inv-cong*:
assumes $r: ((a, b), (a', b')) \in \text{frac-rel}$ **and** $\text{anz}: a \neq 0$
shows $((b, a), (b', a')) \in \text{frac-rel}$
 $\langle \text{proof} \rangle$

66.3.3 The *Class*-formulas

Key auxiliary: if a class is chosen from a fraction, the SOME-representative is equivalent to the given fraction.

lemma *some-rep-equiv*:
assumes $ab: (a, b) \in \text{frac-set}$
defines $p \equiv \text{SOME } p. p \in \text{frac.Class } (a, b) \wedge p \in \text{frac-set}$
shows $(p, (a, b)) \in \text{frac-rel}$ **and** $p \in \text{frac-set}$
 $\langle \text{proof} \rangle$

lemma *frac-add-Class*:
assumes $ab: (a, b) \in \text{frac-set}$ **and** $cd: (c, d) \in \text{frac-set}$
shows $\text{frac.Class } (a, b) [+] \text{frac.Class } (c, d) = \text{frac.Class } (a \cdot d + c \cdot b, b \cdot d)$
 $\langle \text{proof} \rangle$

lemma *frac-mult-Class*:
assumes $ab: (a, b) \in \text{frac-set}$ **and** $cd: (c, d) \in \text{frac-set}$
shows $\text{frac.Class } (a, b) [\cdot] \text{frac.Class } (c, d) = \text{frac.Class } (a \cdot c, b \cdot d)$
 $\langle \text{proof} \rangle$

lemma *frac-neg-Class*:
assumes $ab: (a, b) \in \text{frac-set}$
shows $[-] \text{frac.Class } (a, b) = \text{frac.Class } (- a, b)$
 $\langle \text{proof} \rangle$

lemma *frac-inv-Class*:
assumes $ab: (a, b) \in \text{frac-set}$ **and** $\text{anz}: a \neq 0$
shows $\text{frac-inv } (\text{frac.Class } (a, b)) = \text{frac.Class } (b, a)$
 $\langle \text{proof} \rangle$

66.3.4 Closure under the operations

lemma *frac-add-closed* [*intro*, *simp*]:
 $A \in \text{Fractions} \implies B \in \text{Fractions} \implies A [+] B \in \text{Fractions}$
 $\langle \text{proof} \rangle$

lemma *frac-mult-closed* [*intro*, *simp*]:
 $A \in \text{Fractions} \implies B \in \text{Fractions} \implies A [\cdot] B \in \text{Fractions}$

<proof>

lemma *frac-neg-closed* [*intro*, *simp*]: $A \in \text{Fractions} \implies [-] A \in \text{Fractions}$
<proof>

66.4 The field of fractions

We assemble the additive abelian group and the multiplicative monoid as separate top-level lemmas, then compose them via *Ring.intro* to obtain the ring structure. This avoids taking on the (locale-qualified) *Monoid.invertible* obligation mid-interpretation: we discharge it once, at the level of *GroupI*, which takes the raw $\exists v \in G. u \cdot v = 1 \wedge v \cdot u = 1$ form directly.

66.4.1 The additive abelian group of *Fractions*

The additive group. Associativity, unit, invertibility, and commutativity are established on fraction representatives. Invertibility supplies the explicit inverse *Class*($- a, b$) of *Class*(a, b).

lemma *Frac-add-group*: *Group Fractions* ([+]) [0]
<proof>

lemma *Frac-add-commutative*: $\llbracket A \in \text{Fractions}; B \in \text{Fractions} \rrbracket \implies A [+] B = B [+] A$
<proof>

lemma *Frac-add-commutative-monoid*: *commutative-monoid Fractions* ([+]) [0]
<proof>

lemma *Frac-abelian-group*: *Abelian-Group Fractions* ([+]) [0]
<proof>

66.4.2 The multiplicative monoid of *Fractions*

lemma *Frac-mult-monoid*: *Monoid Fractions* ([·]) [1]
<proof>

66.4.3 Distributivity and the ring axiom

lemma *Frac-Ring-axioms*: *Ring-axioms Fractions* ([+]) ([·])
<proof>

Assemble the additive abelian group, multiplicative monoid, and the distributive laws into the *Ring* interpretation.

interpretation *Frac*: *Ring Fractions* ([+]) ([·]) [0] [1]
<proof>

interpretation *Frac*: *commutative-ring Fractions* ([+]) ([·]) [0] [1]

⟨proof⟩

interpretation *Frac: Field Fractions* ([+]) ([·]) [0] [1]
 ⟨proof⟩

66.5 The canonical embedding $R \rightarrow \text{Fractions}$

The canonical map $a \mapsto \langle \text{Class}(a, \mathbf{1}) \rangle$ embeds R into its field of fractions. It is an injective ring homomorphism. Together with the field structure, this is the universal embedding of an integral domain into a field: every nonzero element of R acquires a multiplicative inverse in *Fractions*.

definition *to-frac* :: 'a $\Rightarrow$ ('a $\times$ 'a) set
where *to-frac* = ($\lambda a \in R.$ *frac.Class* ($a, \mathbf{1}$))

lemma *to-frac-apply* [*simp*]: $a \in R \implies \text{to-frac } a = \text{frac.Class } (a, \mathbf{1})$
 ⟨proof⟩

lemma *to-frac-undefined* [*simp*]: $a \notin R \implies \text{to-frac } a = \text{undefined}$
 ⟨proof⟩

lemma *to-frac-pair*: $a \in R \implies (a, \mathbf{1}) \in \text{frac-set}$
 ⟨proof⟩

lemma *to-frac-in-Fractions* [*intro*, *simp*]: $a \in R \implies \text{to-frac } a \in \text{Fractions}$
 ⟨proof⟩

lemma *to-frac-zero*: *to-frac* $\mathbf{0} = [\mathbf{0}]$ ⟨proof⟩

lemma *to-frac-one*: *to-frac* $\mathbf{1} = [\mathbf{1}]$ ⟨proof⟩

lemma *to-frac-add*:
assumes $a: a \in R$ **and** $b: b \in R$
shows *to-frac* ($a + b$) = *to-frac* a [+] *to-frac* b
 ⟨proof⟩

lemma *to-frac-mult*:
assumes $a: a \in R$ **and** $b: b \in R$
shows *to-frac* ($a \cdot b$) = *to-frac* a [·] *to-frac* b
 ⟨proof⟩

lemma *to-frac-inj*:
assumes $a: a \in R$ **and** $b: b \in R$ **and** *eq*: *to-frac* $a = \text{to-frac } b$
shows $a = b$
 ⟨proof⟩

interpretation *to-frac: ring-homomorphism to-frac R addition* (·) $\mathbf{0} \mathbf{1}$
Fractions ([+]) ([·]) [0] [1]
 ⟨proof⟩

end

end

67 The polynomial ring over a field is a Euclidean domain

theory *Poly-Divisibility*

imports *Poly-Ring Divisibility-Theory Poly-Ideal*

begin

Over a field F , the polynomial ring $F[X]$ is a Euclidean domain, with the polynomial degree as Euclidean function: this is exactly the division algorithm of *Poly-Ring* (*Field.poly-divide*) packaged into the abstract *Euclidean-Domain* of *Divisibility-Theory*. The interpretation is the bridge that endows $F[X]$ — for free — with the whole divisibility tower: it is a principal ideal domain and a factorial (unique factorization) domain, in which irreducible polynomials are prime and factorizations into irreducibles exist and are unique up to the number of factors.

context *Field*

begin

The carrier-set polynomial ring satisfies the abstract Euclidean-domain interface. This named theorem keeps the division-algorithm proof reusable by the exported sublocale below.

theorem *poly-euclidean-domain:*

Euclidean-Domain poly-carrier $(\oplus_P) (\otimes_P) \mathbf{0}_P \mathbf{1}_P$ *degree*
<proof>

end

sublocale *Field* $\subseteq$ *FX: Euclidean-Domain*

poly-carrier $(\oplus_P) (\otimes_P) \mathbf{0}_P \mathbf{1}_P$ *degree*
<proof>

context *Field*

begin

Consequences, transported from the abstract tower. Every nonzero non-unit polynomial factors into irreducibles (*FX.factorization-exists*), an irreducible polynomial is prime (*FX.irreducible-imp-prime*), and any two prime factorizations of associated polynomials have the same length (*FX.prime-factorization-length-unique*).

theorem *poly-factorization-exists:*

$\llbracket p \in \text{poly-carrier}; p \neq \mathbf{0}_P; \neg \text{FX.is-unit } p \rrbracket \implies \exists fs. \text{FX.factorization } fs \ p$
<proof>

theorem *poly-irreducible-imp-prime*:
 $FX.irreducible\text{-}elem\ p \implies FX.prime\text{-}elem\ p$
 ⟨proof⟩

end

Poly-Divisibility interprets the carrier-set polynomial ring as an abstract Euclidean domain, while *Poly-Ideal* exposes the same multiples through *Field.poly-dvd* and *Field.poly-pideal*. The two interfaces are intentionally kept in their original theories so that each remains independently usable. This theory is the explicit seam between them.

context *Field*
begin

The concrete polynomial divisibility relation and the abstract relation differ only by the orientation of the product. Commutativity of the polynomial ring supplies the bridge.

lemma *FX-divides-iff-poly-dvd*:
assumes $d: d \in poly\text{-}carrier$
shows $FX.divides\ d\ x \longleftrightarrow poly\text{-}dvd\ d\ x$
 ⟨proof⟩

The concrete principal ideal and the abstract divisibility relation now have a common polynomial-level formulation.

lemma *FX-divides-iff-mem-poly-pideal*:
assumes $d: d \in poly\text{-}carrier$
shows $FX.divides\ d\ x \longleftrightarrow x \in poly\text{-}pideal\ d$
 ⟨proof⟩

lemma *poly-dvd-iff-mem-FX-principal*:
assumes $d: d \in poly\text{-}carrier$ **and** $x: x \in poly\text{-}carrier$
shows $poly\text{-}dvd\ d\ x \longleftrightarrow x \in FX.principal\text{-}ideal\ d$
 ⟨proof⟩

lemma *FX-divides-iff-pmod-zero*:
assumes $d: d \in poly\text{-}carrier$ **and** $dnz: d \neq 0_P$
and $x: x \in poly\text{-}carrier$
shows $FX.divides\ d\ x \longleftrightarrow pmod\ x\ d = 0_P$
 ⟨proof⟩

67.1 Units and irreducibility across the seam

The same seam again one notion up. *poly-unit* spells out invertibility in the polynomial ring directly, while *FX.is-unit* is the abstract *Monoid.invertible* at that ring. *They are not quite the same predicate*: *Monoid.invertible* does not require its argument to lie in the carrier, whereas *poly-unit* does, so the two agree only on *poly-carrier*. That is enough for the notion actually wanted, because

poly-irreducible and *FX.irreducible-elem* both demand carrier membership outright and apply their unit test only to carrier elements. Their agreement is what lets a factor produced by *poly-factorization-exists* be fed to *poly-quotient-field*.

lemma *FX-is-unit-iff-poly-unit*:

assumes $p: p \in \text{poly-carrier}$ **shows** $\text{FX.is-unit } p \longleftrightarrow \text{poly-unit } p$
 $\langle \text{proof} \rangle$

lemma *FX-irreducible-iff-poly-irreducible*:

$\text{FX.irreducible-elem } p \longleftrightarrow \text{poly-irreducible } p$
 $\langle \text{proof} \rangle$

Every polynomial that is neither zero nor a unit has an irreducible factor: factor it and take the head of the list, which is nonempty because the empty product is $\mathbf{1}_P$. This is the form in which the adjunction of a root consumes factorization — a root of an irreducible factor of p is a root of p — and it is stated with *poly-irreducible* so that *poly-quotient-field* applies directly.

theorem *exists-poly-irreducible-factor*:

assumes $p: p \in \text{poly-carrier}$ **and** $\text{pnz}: p \neq \mathbf{0}_P$ **and** $\text{pnu}: \neg \text{poly-unit } p$
obtains q **where** $\text{poly-irreducible } q$ **and** $\text{poly-dvd } q \ p$
 $\langle \text{proof} \rangle$

Each member of a list divides the product of the list. Needed to feed a single irreducible factor of a factorisation to the adjunction of a root.

lemma *poly-dvd-poly-prod*:

assumes $\text{fs}: \bigwedge g. g \in \text{set fs} \implies g \in \text{poly-carrier}$ **and** $f: f \in \text{set fs}$
shows $\text{poly-dvd } f \ (\text{poly-prod fs})$
 $\langle \text{proof} \rangle$

67.2 The field of remainders modulo an irreducible polynomial

Kronecker's theorem in *normal-form* presentation: rather than the cosets of (p) , take the polynomials of degree below p , with addition unchanged and multiplication followed by reduction. This is the same field as *poly-quotient-field* exhibits, but its carrier is a set of polynomials rather than a set of cosets, so it stays inside the coefficient type. That is what a construction adjoining roots inside a fixed ambient type needs, and it is also the form in which one computes.

Invertibility is the only substantial part, and it does not need gcds: a nonzero remainder is not a multiple of p , so it lies outside the maximal ideal (p) , and extending that ideal by it gives the whole ring — which is to say $\mathbf{1}_P$ is a combination $m \oplus_P r \otimes_P S$ with m a multiple of p . Reducing that equation is exactly the inverse.

theorem *remainder-field*:

assumes $\text{irr}: \text{poly-irreducible } p$
shows $\text{Field } (\text{low-poly } (\text{degree } p)) \ (\oplus_P) \ (\lambda S \ T. \text{pmod } (S \otimes_P T) \ p) \ \mathbf{0}_P \ \mathbf{1}_P$
 $\langle \text{proof} \rangle$

The adjoined root really is a root. Evaluating in the field of remainders, at the variable and with the coefficients embedded as constants, reduces the polynomial modulo the modulus. So the class of q is the value of q at the variable, and when the modulus divides q that value is zero.

Reduction is a ring homomorphism from $F[X]$ onto the remainders, by *pmod-add* and *pmod-mult*, and it fixes the variable and the constants because those are already of low degree — which is where $2 \leq \text{degree } p$ is needed. So *eval-hom* applies, and the source evaluation is the polynomial itself by *eval-const-lift-at-var*.

theorem *eval-remainder-at-var*:

assumes *irr*: *poly-irreducible* p **and** *d2*: $2 \leq \text{degree } p$ **and** $q: q \in \text{poly-carrier}$

shows $\text{Ring.eval } (\text{low-poly } (\text{degree } p)) \ (\oplus_P) \ (\lambda S \ T. \text{ pmod } (S \otimes_P T) \ p)$

$\mathbf{0}_P \ \mathbf{1}_P \ X_P \ (\lambda k. \text{ poly-const } (q \ k)) = \text{ pmod } q \ p$

<proof>

end

end

68 Polynomials in an Indexed Family of Indeterminates

theory *Indexed-Poly*

imports *Poly-Ring HOL-Library.Multiset*

begin

Poly-Ring builds $R[X]$ from finite-support functions $\text{nat} \Rightarrow 'a$, the exponent of the single indeterminate being the index. Here the indeterminates form an arbitrary family, indexed by a type $'i$, and a monomial is therefore not a single exponent but a *multiset* over $'i$: the monomial $X_i^2 X_j$ is $\{\#i, i, j\}$. A polynomial is a finite-support function $'i \text{ multiset} \Rightarrow 'a$ with coefficients in the carrier, so the representation choice is the same one *Poly-Ring* makes, with the exponent *nat* replaced by a monomial. Addition is pointwise and multiplication is the convolution $(p \otimes q) \ m = (\sum n \subseteq\# \ m. p \ n \cdot q \ (m - n))$, the sum running over the submultisets of m — the monomial factorisations $m = n + (m - n)$ — exactly as the univariate convolution runs over $\{..k\}$, the factorisations $k = i + (k - i)$.

Why an indexed family, rather than iterating *Poly-Ring*. Iteration gives $R[X][Y]$ and so on, which is *finitely* many indeterminates. The construction of an algebraic closure needs one indeterminate for every polynomial over the base ring, all present at once, so that a maximal extension has room for a root of each. That is what this theory supplies.

Three facts used below — *fincomp-Sigma*, *fincomp-mult-distrib-left* and *fincomp-mult-distrib-right* — live in *Poly-Ring* but are generic in the index type and say nothing about polynomials.

68.1 Submultisets

The convolution sums over the submultisets of a monomial, so that index set must be finite. Each submultiset of *add-mset* x m either omits x , and is a submultiset of m , or contains it, and is x added to one.

lemma *finite-submultisets*: $\text{finite } \{n. n \subseteq\# (m :: 'i \text{ multiset})\}$
 $\langle \text{proof} \rangle$

Monomial arithmetic for the convolution, all at the level of coefficient counts. The last two are the identities that make the associativity reindexing below a bijection.

lemma *msub-diff-mono*: $\llbracket a \subseteq\# b; b \subseteq\# m \rrbracket \implies b - a \subseteq\# m - a$
 $\langle \text{proof} \rangle$

lemma *msub-add-closed*: $\llbracket a \subseteq\# m; b \subseteq\# m - a \rrbracket \implies a + b \subseteq\# m$
 $\langle \text{proof} \rangle$

lemma *msub-diff-diff*: $\llbracket a \subseteq\# b; b \subseteq\# m \rrbracket \implies m - a - (b - a) = m - b$
 $\langle \text{proof} \rangle$

Complementation is an involution of the submultisets of a monomial; this is what makes the convolution symmetric when the coefficients commute.

lemma *msub-double-diff*: $n \subseteq\# m \implies m - (m - n) = n$
 $\langle \text{proof} \rangle$

context *Ring*
begin

68.2 Carrier and coefficients

The polynomials over R in the indeterminates indexed by $'i$: finite support on monomials, coefficients in the carrier.

definition *ipoly-carrier* :: $('i \text{ multiset} \Rightarrow 'a) \text{ set}$
where *ipoly-carrier* $\equiv \{p. \text{finite } \{m. p \ m \neq \mathbf{0}\} \wedge (\forall m. p \ m \in R)\}$

lemma *ipoly-carrierI*:
 $\llbracket \text{finite } \{m. p \ m \neq \mathbf{0}\}; \bigwedge m. p \ m \in R \rrbracket \implies p \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

lemma *ipoly-carrier-finite*: $p \in \text{ipoly-carrier} \implies \text{finite } \{m. p \ m \neq \mathbf{0}\}$
 $\langle \text{proof} \rangle$

lemma *ipoly-carrier-coeff-closed*: $p \in \text{ipoly-carrier} \implies p \ m \in R$
 $\langle \text{proof} \rangle$

68.3 Operations

definition *ipoly-zero* :: $'i \text{ multiset} \Rightarrow 'a \ (\langle \mathbf{0}_I \rangle)$
where *ipoly-zero* $\equiv (\lambda m. \mathbf{0})$

The unit is the constant $\mathbf{1}$, carried by the empty monomial.

definition *ipoly-one* :: $'i \text{ multiset} \Rightarrow 'a \ (\langle \mathbf{1}_I \rangle)$
where *ipoly-one* $\equiv (\lambda m. \text{if } m = \{\#\} \text{ then } \mathbf{1} \text{ else } \mathbf{0})$

definition *ipoly-add* :: $('i \text{ multiset} \Rightarrow 'a) \Rightarrow ('i \text{ multiset} \Rightarrow 'a) \Rightarrow ('i \text{ multiset} \Rightarrow 'a)$
(infixl $\langle \oplus_I \rangle$ 65)
where *ipoly-add* $p \ q \equiv (\lambda m. p \ m + q \ m)$

definition *ipoly-neg* :: $('i \text{ multiset} \Rightarrow 'a) \Rightarrow ('i \text{ multiset} \Rightarrow 'a) \ (\langle \ominus_I \rightarrow [66] \ 65)$
where *ipoly-neg* $p \equiv (\lambda m. - p \ m)$

The convolution: a coefficient of the product sums over the factorisations $m = n + (m - n)$ of the monomial.

definition *ipoly-mult* :: $('i \text{ multiset} \Rightarrow 'a) \Rightarrow ('i \text{ multiset} \Rightarrow 'a) \Rightarrow ('i \text{ multiset} \Rightarrow 'a)$
(infixl $\langle \otimes_I \rangle$ 70)
where *ipoly-mult* $p \ q \equiv (\lambda m. \text{additive.fincomp } (\lambda n. p \ n \cdot q \ (m - n)) \ \{n. n \subseteq \# \ m\})$

68.4 Closure under the operations

lemma *ipoly-zero-closed*: $\mathbf{0}_I \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

lemma *ipoly-one-closed*: $\mathbf{1}_I \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

lemma *ipoly-add-closed*:
assumes $p \in \text{ipoly-carrier}$ $q \in \text{ipoly-carrier}$
shows $p \oplus_I q \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

lemma *ipoly-neg-closed*:
assumes $p: p \in \text{ipoly-carrier}$ **shows** $\ominus_I p \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

lemma *ipoly-mult-coeff-closed*:
assumes $p \in \text{ipoly-carrier}$ $q \in \text{ipoly-carrier}$
shows $(p \otimes_I q) \ m \in R$
 $\langle \text{proof} \rangle$

A product coefficient can be nonzero only at a monomial that factors as a support monomial of p times one of q , and there are finitely many such products. This replaces the degree bound used in the univariate case, where the support is bounded by a single number.

lemma *ipoly-mult-support-bound*:

assumes $p: p \in \text{ipoly-carrier}$ **and** $q: q \in \text{ipoly-carrier}$
shows $\{m. (p \otimes_I q) \cdot m \neq \mathbf{0}\} \subseteq (\lambda(a, b). a + b) \cdot (\{m. p \cdot m \neq \mathbf{0}\} \times \{m. q \cdot m \neq \mathbf{0}\})$
 $\langle \text{proof} \rangle$

lemma *ipoly-mult-closed*:
assumes $p \in \text{ipoly-carrier}$ **and** $q \in \text{ipoly-carrier}$
shows $p \otimes_I q \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

68.5 Additive structure

Addition is pointwise, so the abelian-group laws are those of R coefficientwise.

lemma *ipoly-add-assoc*:
 $\llbracket p \in \text{ipoly-carrier}; q \in \text{ipoly-carrier}; r \in \text{ipoly-carrier} \rrbracket$
 $\implies (p \oplus_I q) \oplus_I r = p \oplus_I (q \oplus_I r)$
 $\langle \text{proof} \rangle$

lemma *ipoly-add-comm*:
 $\llbracket p \in \text{ipoly-carrier}; q \in \text{ipoly-carrier} \rrbracket \implies p \oplus_I q = q \oplus_I p$
 $\langle \text{proof} \rangle$

lemma *ipoly-add-zero*: $p \in \text{ipoly-carrier} \implies \mathbf{0}_I \oplus_I p = p$
 $\langle \text{proof} \rangle$

lemma *ipoly-add-neg*: $p \in \text{ipoly-carrier} \implies (\ominus_I p) \oplus_I p = \mathbf{0}_I$
 $\langle \text{proof} \rangle$

lemma *ipoly-add-neg-right*: $p \in \text{ipoly-carrier} \implies p \oplus_I (\ominus_I p) = \mathbf{0}_I$
 $\langle \text{proof} \rangle$

68.6 Multiplicative unit

Only the empty monomial contributes to a product with the unit.

lemma *ipoly-mult-one-left*:
assumes $p: p \in \text{ipoly-carrier}$
shows $\mathbf{1}_I \otimes_I p = p$
 $\langle \text{proof} \rangle$

lemma *ipoly-mult-one-right*:
assumes $p: p \in \text{ipoly-carrier}$
shows $p \otimes_I \mathbf{1}_I = p$
 $\langle \text{proof} \rangle$

68.7 Distributivity

lemma *ipoly-mult-add-distrib-left*:
assumes $p: p \in \text{ipoly-carrier}$ **and** $q: q \in \text{ipoly-carrier}$ **and** $r: r \in \text{ipoly-carrier}$
shows $p \otimes_I (q \oplus_I r) = (p \otimes_I q) \oplus_I (p \otimes_I r)$

$\langle \text{proof} \rangle$

lemma *ipoly-mult-add-distrib-right*:

assumes $p: p \in \text{ipoly-carrier}$ **and** $q: q \in \text{ipoly-carrier}$ **and** $r: r \in \text{ipoly-carrier}$

shows $(q \oplus_I r) \otimes_I p = (q \otimes_I p) \oplus_I (r \otimes_I p)$

$\langle \text{proof} \rangle$

68.8 Associativity of the convolution

Both sides are the sum of $p \cdot a \cdot q \cdot b \cdot r \cdot c$ over the factorisations $m = a + b + c$. The two nested sums present that set differently — as pairs $(b + a, a)$ on the left and (a, b) on the right — so the proof is a reindexing along $(b, a) \mapsto (a, b - a)$, exactly as in the univariate case, with monomial arithmetic in place of arithmetic on degrees.

lemma *ipoly-mult-assoc*:

fixes $p \ q \ r :: 'i \text{ multiset} \Rightarrow 'a$

assumes $p: p \in \text{ipoly-carrier}$ **and** $q: q \in \text{ipoly-carrier}$ **and** $r: r \in \text{ipoly-carrier}$

shows $(p \otimes_I q) \otimes_I r = p \otimes_I (q \otimes_I r)$

$\langle \text{proof} \rangle$

68.9 The indexed polynomial ring

theorem *ipoly-ring*: *Ring ipoly-carrier* $(\oplus_I) (\otimes_I) \mathbf{0}_I \mathbf{1}_I$

$\langle \text{proof} \rangle$

68.10 Constants and indeterminates

A constant carries its value on the empty monomial. This is the embedding of the coefficient ring, and it is what lets an extension of R be built inside this ring.

definition *ipoly-const* $:: 'a \Rightarrow ('i \text{ multiset} \Rightarrow 'a)$

where *ipoly-const* $a \equiv (\lambda m. \text{if } m = \{\#\} \text{ then } a \text{ else } \mathbf{0})$

lemma *ipoly-const-coeff* [simp]: *ipoly-const* $a \ \{\#\} = a$

$\langle \text{proof} \rangle$

lemma *ipoly-const-closed*: $a \in R \implies \text{ipoly-const } a \in \text{ipoly-carrier}$

$\langle \text{proof} \rangle$

lemma *ipoly-const-zero* [simp]: *ipoly-const* $\mathbf{0} = \mathbf{0}_I$

$\langle \text{proof} \rangle$

lemma *ipoly-const-one* [simp]: *ipoly-const* $\mathbf{1} = \mathbf{1}_I$

$\langle \text{proof} \rangle$

lemma *ipoly-const-add*:

assumes $a \in R$ **and** $b \in R$

shows *ipoly-const* $(a + b) = (\text{ipoly-const } a :: 'i \text{ multiset} \Rightarrow 'a) \oplus_I \text{ipoly-const } b$

$\langle proof \rangle$

lemma *ipoly-const-mult*:

assumes $a \in R \ b \in R$

shows $ipoly-const \ (a \cdot b) = (ipoly-const \ a :: 'i \ multiset \Rightarrow 'a) \otimes_I ipoly-const \ b$

$\langle proof \rangle$

lemma *ipoly-const-inj-on*: $inj-on \ (ipoly-const :: 'a \Rightarrow 'i \ multiset \Rightarrow 'a) \ R$

$\langle proof \rangle$

The indeterminate X_i is the monomial $\{\#i\#$ with coefficient $\mathbf{1}$. Distinct indices give distinct indeterminates whenever the ring is nontrivial: note that *Ring* does not itself assume $\mathbf{1} \neq \mathbf{0}$.

definition *ivar* :: $'i \Rightarrow ('i \ multiset \Rightarrow 'a)$

where $ivar \ i \equiv (\lambda m. \text{if } m = \{\#i\# \} \text{ then } \mathbf{1} \text{ else } \mathbf{0})$

lemma *ivar-closed*: $ivar \ i \in ipoly-carrier$

$\langle proof \rangle$

lemma *ivar-inj*:

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

shows $inj \ (ivar :: 'i \Rightarrow 'i \ multiset \Rightarrow 'a)$

$\langle proof \rangle$

68.11 Indeterminates a polynomial does not involve

$\mathcal{P}$ does not involve the indeterminate j when no monomial containing j carries a nonzero coefficient. An index that *no* element of an extension involves is what supplies room for a new root: adjoining one is then guaranteed not to disturb what is there already. This replaces the cardinality argument that a construction over an abstract type of labels would need, and is the reason for choosing indexed polynomials as the ambient type.

definition *index-free* :: $('i \ multiset \Rightarrow 'a) \Rightarrow 'i \Rightarrow bool$

where $index-free \ \mathcal{P} \ j \equiv (\forall m. j \in \# \ m \longrightarrow \mathcal{P} \ m = \mathbf{0})$

lemma *index-freeI*: $(\bigwedge m. j \in \# \ m \Longrightarrow \mathcal{P} \ m = \mathbf{0}) \Longrightarrow index-free \ \mathcal{P} \ j$

$\langle proof \rangle$

lemma *index-freeD*: $\llbracket index-free \ \mathcal{P} \ j; j \in \# \ m \rrbracket \Longrightarrow \mathcal{P} \ m = \mathbf{0}$

$\langle proof \rangle$

Constants involve no indeterminate at all.

lemma *index-free-const* [simp]: $index-free \ (ipoly-const \ a) \ j$

$\langle proof \rangle$

lemma *index-free-zero* [simp]: $index-free \ \mathbf{0}_I \ j$

$\langle proof \rangle$

The indeterminate *ivar j* does involve *j*, provided the coefficient ring is nontrivial
 — *Ring* does not itself assume $\mathbf{1} \neq \mathbf{0}$.

lemma *not-index-free-ivar*:
 assumes $\mathbf{1} \neq \mathbf{0}$ shows $\neg \text{index-free } (\text{ivar } j) \ j$
 $\langle \text{proof} \rangle$

68.12 Monomials

The monomial *imonom a n* carries the coefficient *a* on the monomial *n*. Constants and indeterminates are the special cases $n = \{\#\}$ and $a = \mathbf{1}$ with *n* a singleton.

definition *imonom* :: $'a \Rightarrow 'i \text{ multiset} \Rightarrow ('i \text{ multiset} \Rightarrow 'a)$
 where *imonom a n* $\equiv (\lambda m. \text{if } m = n \text{ then } a \text{ else } \mathbf{0})$

lemma *imonom-apply [simp]*: *imonom a n n* = *a*
 $\langle \text{proof} \rangle$

lemma *imonom-closed*: $a \in R \implies \text{imonom } a \ n \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

lemma *ipoly-const-eq-imonom*: *ipoly-const a* = *imonom a* $\{\#\}$
 $\langle \text{proof} \rangle$

lemma *ivar-eq-imonom*: *ivar j* = *imonom* $\mathbf{1} \ \{\#j\#\}$
 $\langle \text{proof} \rangle$

lemma *imonom-mult*:
 fixes *n n'* :: $'i \text{ multiset}$
 assumes *a*: $a \in R$ and *b*: $b \in R$
 shows *imonom a n* $\otimes_I$ *imonom b n'* = *imonom* $(a \cdot b) \ (n + n')$
 $\langle \text{proof} \rangle$

lemma *imonom-shift*:
 fixes *n m* :: $'i \text{ multiset}$
 assumes *c*: $c \in \text{ipoly-carrier}$
 shows $(c \otimes_I \text{imonom } \mathbf{1} \ n) \ m = (\text{if } n \subseteq\# m \text{ then } c \ (m - n) \text{ else } \mathbf{0})$
 $\langle \text{proof} \rangle$

A power of an indeterminate is the monomial with that exponent.

lemma *ivar-pow*:
 fixes *j* :: $'i$
 shows *Ring.rpow* $(\otimes_I) \ \mathbf{1}_I \ (\text{ivar } j) \ k = \text{imonom } \mathbf{1} \ (\text{replicate-mset } k \ j)$
 $\langle \text{proof} \rangle$

68.13 Reading a polynomial off its value at an indeterminate

The point of the monomial lemmas above. A sum $\sum_k c_k X_j^k$ whose coefficients do not involve X_j determines those coefficients: its value on a monomial m is the coefficient indexed by the multiplicity of j in m , evaluated at what remains of m . So such a sum is injective in the coefficients, which is what makes the realisation of a quotient $M[X]/(Q)$ inside this ring faithful.

lemma *ipoly-add-cmonoid: commutative-monoid ipoly-carrier $(\oplus_I) \mathbf{0}_I$*
<proof>

Addition is pointwise, so a finite sum may be evaluated coefficientwise.

lemma *ipoly-fincomp-apply:*
fixes $m :: 'i \text{ multiset}$
assumes $f: \bigwedge k. f\ k \in \text{ipoly-carrier}$
shows $\text{commutative-monoid.fincomp ipoly-carrier } (\oplus_I) \mathbf{0}_I\ f\ A\ m = \text{additive.fincomp } (\lambda k. f\ k\ m)\ A$
<proof>

lemma *replicate-mset-subseteq-count: replicate-mset $k\ j \subseteq\# m \longleftrightarrow k \leq \text{count } m\ j$*
<proof>

The closed form: only the term whose exponent matches the multiplicity of j in m survives, because every earlier coefficient is evaluated at a monomial that still involves j , where it vanishes.

theorem *ieval-ivar:*
fixes $j :: 'i$ *and* $m :: 'i \text{ multiset}$
assumes $c: \bigwedge k. c\ k \in \text{ipoly-carrier}$ *and* *free:* $\bigwedge k. \text{index-free } (c\ k)\ j$
shows $\text{commutative-monoid.fincomp ipoly-carrier } (\oplus_I) \mathbf{0}_I$
 $(\lambda k. c\ k \otimes_I \text{Ring.rpow } (\otimes_I) \mathbf{1}_I\ (\text{ivar } j)\ k)\ \{..d\}\ m$
 $= (\text{if } \text{count } m\ j \leq d \text{ then } c\ (\text{count } m\ j)\ (m - \text{replicate-mset } (\text{count } m\ j)\ j) \text{ else } \mathbf{0})$
<proof>

What the closed form is for: the coefficients can be recovered, so a polynomial in $\text{ivar } j$ with index-free coefficients determines them. Reading *ieval-ivar* at the monomial $n + \text{replicate-mset } k\ j$ isolates the k th coefficient at n — provided n itself avoids j , and where it does not, both coefficients vanish anyway. This is the injectivity that lets a quotient be realised by normal forms: distinct remainders go to distinct elements of the ambient ring.

lemma *ieval-ivar-inj:*
fixes $j :: 'i$ *and* $D\ k :: \text{nat}$
assumes $c: \bigwedge k. c\ k \in \text{ipoly-carrier}$ *and* *cfree:* $\bigwedge k. \text{index-free } (c\ k)\ j$
and $d: \bigwedge k. d\ k \in \text{ipoly-carrier}$ *and* *dfree:* $\bigwedge k. \text{index-free } (d\ k)\ j$
and $\text{eq: commutative-monoid.fincomp ipoly-carrier } (\oplus_I) \mathbf{0}_I$
 $(\lambda k. c\ k \otimes_I \text{Ring.rpow } (\otimes_I) \mathbf{1}_I\ (\text{ivar } j)\ k)\ \{..D\}$
 $= \text{commutative-monoid.fincomp ipoly-carrier } (\oplus_I) \mathbf{0}_I$

```

      (λk. d k ⊗I Ring.rpow (⊗I) 1I (ivar j) k) {..D}
    and k: k ≤ D
    shows c k = d k
  ⟨proof⟩

end

```

69 Polynomials over a Commutative Ring

```

context commutative-ring
begin

```

When the coefficients commute so do the polynomials: the convolution is symmetric under the involution $n \mapsto m - n$ of the submultisets of m .

```

lemma ipoly-mult-comm:
  fixes p q :: 'i multiset ⇒ 'a
  assumes p: p ∈ ipoly-carrier and q: q ∈ ipoly-carrier
  shows p ⊗I q = q ⊗I p
  ⟨proof⟩

```

```

theorem ipoly-commutative-ring:
  commutative-ring (ipoly-carrier :: ('i multiset ⇒ 'a) set) (⊕I) (⊗I) 0I 1I
  ⟨proof⟩

```

```

end

```

70 The Univariate Case, as a Check on the Convolution

With a single indeterminate — index type *unit* — a monomial is determined by its size, so the submultisets of X^k are exactly X^i for $i \leq k$, and the indexed convolution must reduce to the univariate one of *Poly-Ring*. This is an independent check on both the index set of the convolution and the placement of $m - n$ within it: a sum over the wrong factorisations of a monomial would still build a ring, and would still satisfy every law proved above, but it would not agree with *Poly-Ring* here.

```

lemma unit-mset-eq-replicate: (n :: unit multiset) = replicate-mset (count n ()) ()
  ⟨proof⟩

```

```

lemma submultisets-replicate-unit:
  {n :: unit multiset. n ⊆# replicate-mset k ()} = (λi. replicate-mset i ()) ‘ {..k}
  ⟨proof⟩

```

```

lemma replicate-unit-diff:
  replicate-mset k () - replicate-mset i () = replicate-mset (k - i) ()
  ⟨proof⟩

```

lemma *inj-on-replicate-unit*: *inj-on* ($\lambda i. \text{replicate-mset } i \ ()$) $\{..k\}$
 $\langle \text{proof} \rangle$

context *Ring*
begin

theorem *ipoly-mult-unit-index*:
fixes $p \ q :: \text{unit multiset} \Rightarrow 'a$
assumes $p: p \in \text{ipoly-carrier}$ **and** $q: q \in \text{ipoly-carrier}$
shows $(p \otimes_I q) (\text{replicate-mset } k \ ())$
 $= ((\lambda i. p (\text{replicate-mset } i \ ())) \otimes_P (\lambda i. q (\text{replicate-mset } i \ ()))) k$
 $\langle \text{proof} \rangle$

end

end

71 Extensions of a Field inside the Indexed Polynomial Ring

theory *Closure-Extension*
imports *Indexed-Poly Poly-Divisibility HOL.Zorn*
begin

Groundwork for the existence of an algebraic closure. Every extension needed for that is generated by adjoining roots of polynomials over the base field R , so all of them can be built inside *one* fixed type: the indexed polynomials of *Indexed-Poly* whose indeterminates are indexed by pairs (P, i) — a polynomial over R together with a natural number, so that each polynomial has as many indeterminates available as it could have roots.

Fixing the type in advance is what makes Zorn's lemma applicable: the extensions then form a *set*, which can be ordered and exhausted. The alternative, iterating a quotient construction to adjoin one root at a time, changes the type at every step — a quotient of a ring over $'a$ has carrier of type $'a \text{ set}$ — and so cannot be iterated by induction at all.

An extension is a triple $(K, \oplus, \otimes)$: a carrier with its own addition and multiplication. The operations must be allowed to vary, because an extension realises a quotient $M[X]/(Q)$ concretely, by normal forms, and the modulus Q differs from step to step. The zero and the unit, by contrast, are the same throughout: the constants $\mathbf{0}_I$ and $\mathbf{1}_I$.

Why the operations are required to be undefined off the carrier. Two extensions with the same carrier and the same operations *on* that carrier are the same extension mathematically, but need not be equal as triples if the operations disagree elsewhere — and then the ordering below fails antisymmetry, so it is not a partial order and Zorn does not apply. HOL-Algebra repairs this after the

fact with a *law-restrict* operator and a family of lemmas transporting field-ness, homomorphisms and isomorphisms across it. Here it is part of what an extension is, which costs one condition and no lemmas: in the carrier-set architecture the operations are parameters that can be constrained, rather than record fields that are total by construction.

type-synonym $'a \text{ root-index} = (\text{nat} \Rightarrow 'a) \times \text{nat}$
type-synonym $'a \text{ ipoly} = 'a \text{ root-index multiset} \Rightarrow 'a$
type-synonym $'a \text{ iop} = 'a \text{ ipoly} \Rightarrow 'a \text{ ipoly} \Rightarrow 'a \text{ ipoly}$
type-synonym $'a \text{ iext} = 'a \text{ ipoly set} \times 'a \text{ iop} \times 'a \text{ iop}$

abbreviation $\text{ecarrier} :: 'a \text{ iext} \Rightarrow 'a \text{ ipoly set}$ **where** $\text{ecarrier } E \equiv \text{fst } E$
abbreviation $\text{eadd} :: 'a \text{ iext} \Rightarrow 'a \text{ iop}$ **where** $\text{eadd } E \equiv \text{fst } (\text{snd } E)$
abbreviation $\text{emult} :: 'a \text{ iext} \Rightarrow 'a \text{ iop}$ **where** $\text{emult } E \equiv \text{snd } (\text{snd } E)$

71.1 Operations defined only on their carrier

An operation is *restricted* to K when it is *undefined* unless both arguments lie in K . This is the condition discussed above; the lemma after it is the only thing the condition is ever used for, and replaces HOL-Algebra's *law-restrict* apparatus.

definition $\text{op-restricted} :: 'a \text{ ipoly set} \Rightarrow 'a \text{ iop} \Rightarrow \text{bool}$
where $\text{op-restricted } K f \longleftrightarrow (\forall x y. (x \notin K \vee y \notin K) \longrightarrow f x y = \text{undefined})$

lemma op-restrictedI :
 $(\bigwedge x y. x \notin K \vee y \notin K \Longrightarrow f x y = \text{undefined}) \Longrightarrow \text{op-restricted } K f$
 $\langle \text{proof} \rangle$

lemma op-restrictedD :
 $\llbracket \text{op-restricted } K f; x \notin K \vee y \notin K \rrbracket \Longrightarrow f x y = \text{undefined}$
 $\langle \text{proof} \rangle$

Two operations restricted to the same carrier and agreeing on it are equal.

lemma restricted-op-eqI :
assumes $f: \text{op-restricted } K f$ **and** $g: \text{op-restricted } K g$
and $\text{eq}: \bigwedge x y. \llbracket x \in K; y \in K \rrbracket \Longrightarrow f x y = g x y$
shows $f = g$
— Two applications of *ext*, not *intro*: the result type $'a \text{ ipoly}$ is itself a function type, so a greedy *intro* strips a third argument.
 $\langle \text{proof} \rangle$

context Ring
begin

71.2 Lifted polynomials

A polynomial over R read as a polynomial over the ambient ring, coefficientwise. Where HOL-Algebra maps a coefficient *list* through its constant embedding, a polynomial here is already a function on exponents, so the lift is a composition.

definition $\text{lift-poly} :: (\text{nat} \Rightarrow 'a) \Rightarrow (\text{nat} \Rightarrow 'a \text{ ipoly})$
where $\text{lift-poly } P = (\lambda k. \text{ipoly-const } (P \ k))$

lemma $\text{lift-poly-apply} \ [\text{simp}]: \text{lift-poly } P \ k = \text{ipoly-const } (P \ k)$
 $\langle \text{proof} \rangle$

lemma $\text{lift-poly-coeff-closed}: P \in \text{poly-carrier} \Longrightarrow \text{lift-poly } P \ k \in \text{ipoly-carrier}$
 $\langle \text{proof} \rangle$

71.3 What it is to be an extension

Evaluation in an extension, at that extension's own operations.

abbreviation $\text{eeval} :: 'a \text{ iext} \Rightarrow 'a \text{ ipoly} \Rightarrow (\text{nat} \Rightarrow 'a \text{ ipoly}) \Rightarrow 'a \text{ ipoly}$
where $\text{eeval } E \equiv \text{Ring.eval } (\text{ecarrier } E) \ (\text{eadd } E) \ (\text{emult } E) \ \mathbf{0}_I \ \mathbf{1}_I$

Evaluation in the *ambient* indexed polynomial ring. A locale prefix is not available here: we are inside *Ring* already, and interpreting the indexed polynomial ring as a *Ring* at this point would be self-referential. An abbreviation costs nothing and hides the five parameters just as well.

abbreviation $\text{ieval} :: 'a \text{ ipoly} \Rightarrow (\text{nat} \Rightarrow 'a \text{ ipoly}) \Rightarrow 'a \text{ ipoly}$
where $\text{ieval} \equiv \text{Ring.eval ipoly-carrier } (\oplus_I) \ (\otimes_I) \ \mathbf{0}_I \ \mathbf{1}_I$

An extension is closed under the roots it mentions: whenever one of its elements involves the indeterminate (P, i) , that indeterminate belongs to the extension, and is a root of P there. This is the condition that forces a *maximal* extension to split every polynomial — an unsplit polynomial leaves some index unused, and adjoining a root at it yields a strictly larger extension.

definition $\text{root-closed} :: 'a \text{ iext} \Rightarrow \text{bool}$
where $\text{root-closed } E \longleftrightarrow$
 $(\forall \mathcal{P} \in \text{ecarrier } E. \forall P \in \text{poly-carrier}. \forall i.$
 $\neg \text{index-free } \mathcal{P} \ (P, i) \longrightarrow$
 $\text{ivar } (P, i) \in \text{ecarrier } E \wedge \text{eeval } E \ (\text{ivar } (P, i)) \ (\text{lift-poly } P) = \mathbf{0}_I)$

lemma root-closedI :
 $(\bigwedge \mathcal{P} \ P \ i. \llbracket \mathcal{P} \in \text{ecarrier } E; P \in \text{poly-carrier}; \neg \text{index-free } \mathcal{P} \ (P, i) \rrbracket$
 $\implies \text{ivar } (P, i) \in \text{ecarrier } E \wedge \text{eeval } E \ (\text{ivar } (P, i)) \ (\text{lift-poly } P) = \mathbf{0}_I)$
 $\implies \text{root-closed } E$
 $\langle \text{proof} \rangle$

lemma root-closed-mem :
 $\llbracket \text{root-closed } E; \mathcal{P} \in \text{ecarrier } E; P \in \text{poly-carrier}; \neg \text{index-free } \mathcal{P} \ (P, i) \rrbracket$
 $\implies \text{ivar } (P, i) \in \text{ecarrier } E$
 $\langle \text{proof} \rangle$

lemma root-closed-eval :
 $\llbracket \text{root-closed } E; \mathcal{P} \in \text{ecarrier } E; P \in \text{poly-carrier}; \neg \text{index-free } \mathcal{P} \ (P, i) \rrbracket$
 $\implies \text{eeval } E \ (\text{ivar } (P, i)) \ (\text{lift-poly } P) = \mathbf{0}_I$
 $\langle \text{proof} \rangle$

An extension of R inside the indexed polynomial ring: a field of indexed polynomials over R that contains the constants, induces on them the ring operations of R , and is closed under the roots it mentions.

definition *is-extension* :: 'a iext $\Rightarrow$ bool

where *is-extension* $E \longleftrightarrow$

$Field (ecarrier E) (eadd E) (emult E) \mathbf{0}_I \mathbf{1}_I$
 $\wedge ecarrier E \subseteq ipoly-carrier$
 $\wedge (\forall a \in R. ipoly-const a \in ecarrier E)$
 $\wedge (\forall a \in R. \forall b \in R. eadd E (ipoly-const a) (ipoly-const b) = ipoly-const (a + b))$
 $\wedge (\forall a \in R. \forall b \in R. emult E (ipoly-const a) (ipoly-const b) = ipoly-const (a \cdot b))$
 $\wedge root-closed E$
 $\wedge op-restricted (ecarrier E) (eadd E)$
 $\wedge op-restricted (ecarrier E) (emult E)$

definition *extensions* :: 'a iext set

where *extensions* = { $E. is-extension E$ }

lemma *extensions-iff* [simp]: $E \in extensions \longleftrightarrow is-extension E$

$\langle proof \rangle$

The component facts, so that clients never unfold *is-extension*.

lemma *is-extension-field*: $is-extension E \Longrightarrow Field (ecarrier E) (eadd E) (emult E)$

$\mathbf{0}_I \mathbf{1}_I$

$\langle proof \rangle$

lemma *is-extension-subset*: $is-extension E \Longrightarrow ecarrier E \subseteq ipoly-carrier$

$\langle proof \rangle$

lemma *is-extension-carrier-closed*: $\llbracket is-extension E; x \in ecarrier E \rrbracket \Longrightarrow x \in ipoly-carrier$

$\langle proof \rangle$

lemma *is-extension-const-mem*: $\llbracket is-extension E; a \in R \rrbracket \Longrightarrow ipoly-const a \in ecarrier E$

$\langle proof \rangle$

lemma *is-extension-const-add*:

$\llbracket is-extension E; a \in R; b \in R \rrbracket$
 $\Longrightarrow eadd E (ipoly-const a) (ipoly-const b) = ipoly-const (a + b)$

$\langle proof \rangle$

lemma *is-extension-const-mult*:

$\llbracket is-extension E; a \in R; b \in R \rrbracket$
 $\Longrightarrow emult E (ipoly-const a) (ipoly-const b) = ipoly-const (a \cdot b)$

$\langle proof \rangle$

lemma *is-extension-root-closed*: $is-extension E \Longrightarrow root-closed E$

$\langle proof \rangle$

lemma *is-extension-restricted*:

$is-extension\ E \implies op-restricted\ (ecarrier\ E)\ (eadd\ E)$
 $is-extension\ E \implies op-restricted\ (ecarrier\ E)\ (emult\ E)$
 $\langle proof \rangle$

lemma *is-extensionI*:

assumes $Field\ (ecarrier\ E)\ (eadd\ E)\ (emult\ E)\ \mathbf{0}_I\ \mathbf{1}_I$
and $ecarrier\ E \subseteq ipoly-carrier$
and $\bigwedge a. a \in R \implies ipoly-const\ a \in ecarrier\ E$
and $\bigwedge a\ b. \llbracket a \in R; b \in R \rrbracket$
 $\implies eadd\ E\ (ipoly-const\ a)\ (ipoly-const\ b) = ipoly-const\ (a + b)$
and $\bigwedge a\ b. \llbracket a \in R; b \in R \rrbracket$
 $\implies emult\ E\ (ipoly-const\ a)\ (ipoly-const\ b) = ipoly-const\ (a \cdot b)$
and $root-closed\ E$
and $op-restricted\ (ecarrier\ E)\ (eadd\ E)$
and $op-restricted\ (ecarrier\ E)\ (emult\ E)$
shows $is-extension\ E$
 $\langle proof \rangle$

71.4 The ordering by inclusion

One extension extends another when its carrier is contained in the other's and its operations agree with the other's wherever the smaller one is defined.

definition $ext-le :: 'a\ iext \Rightarrow 'a\ iext \Rightarrow bool$

where $ext-le\ E\ F \longleftrightarrow ecarrier\ E \subseteq ecarrier\ F$
 $\wedge (\forall x \in ecarrier\ E. \forall y \in ecarrier\ E.$
 $eadd\ E\ x\ y = eadd\ F\ x\ y \wedge emult\ E\ x\ y = emult\ F\ x\ y)$

lemma *ext-leI*:

$\llbracket ecarrier\ E \subseteq ecarrier\ F;$
 $\bigwedge x\ y. \llbracket x \in ecarrier\ E; y \in ecarrier\ E \rrbracket \implies eadd\ E\ x\ y = eadd\ F\ x\ y;$
 $\bigwedge x\ y. \llbracket x \in ecarrier\ E; y \in ecarrier\ E \rrbracket \implies emult\ E\ x\ y = emult\ F\ x\ y \rrbracket$
 $\implies ext-le\ E\ F$
 $\langle proof \rangle$

lemma *ext-le-carrier*: $ext-le\ E\ F \implies ecarrier\ E \subseteq ecarrier\ F$
 $\langle proof \rangle$

lemma *ext-le-add*: $\llbracket ext-le\ E\ F; x \in ecarrier\ E; y \in ecarrier\ E \rrbracket \implies eadd\ E\ x\ y = eadd\ F\ x\ y$
 $\langle proof \rangle$

lemma *ext-le-mult*: $\llbracket ext-le\ E\ F; x \in ecarrier\ E; y \in ecarrier\ E \rrbracket \implies emult\ E\ x\ y = emult\ F\ x\ y$
 $\langle proof \rangle$

lemma *ext-le-refl*: $ext-le\ E\ E$
 $\langle proof \rangle$

lemma *ext-le-trans*:

assumes EF : *ext-le* E F **and** FG : *ext-le* F G **shows** *ext-le* E G
 ⟨*proof*⟩

Antisymmetry is where restrictedness earns its place: equal carriers and agreement on them settle the operations there, and off the carrier both are *undefined*.

lemma *ext-le-antisym*:
assumes E : *is-extension* E **and** F : *is-extension* F
and EF : *ext-le* E F **and** FE : *ext-le* F E
shows $E = F$
 ⟨*proof*⟩

theorem *ext-partial-order*: *partial-order-on extensions (relation-of ext-le extensions)*
 ⟨*proof*⟩

71.5 Powers and evaluation transfer along the ordering

A power or an evaluation of an element of the smaller extension is the same whether computed there or in the larger one. Nothing beyond ring-ness of the two, carrier inclusion, and agreement of the operations on the smaller carrier is used — that is, exactly *ext-le* — so the same two lemmas serve the union of a chain and the adjunction of a root.

There is no such fact one level down: *commutative-monoid.fincomp* depends on the *carrier* as well as the operation, through its *M-ify*, so *commutative-monoid.fincomp-cong'*, which stays inside one monoid, does not apply. *fincomp-transfer* is what bridges the two carriers.

lemma *ext-le-rpow*:
assumes RE : *Ring* (*ecarrier* E) (*eadd* E) (*emult* E) $\mathbf{0}_I$ $\mathbf{1}_I$
and RF : *Ring* (*ecarrier* F) (*eadd* F) (*emult* F) $\mathbf{0}_I$ $\mathbf{1}_I$
and le : *ext-le* E F **and** a : $a \in \text{ecarrier } E$
shows *Ring.rpow* (*emult* F) $\mathbf{1}_I$ a n = *Ring.rpow* (*emult* E) $\mathbf{1}_I$ a n
 ⟨*proof*⟩

lemma *ext-le-eval*:
assumes RE : *Ring* (*ecarrier* E) (*eadd* E) (*emult* E) $\mathbf{0}_I$ $\mathbf{1}_I$
and RF : *Ring* (*ecarrier* F) (*eadd* F) (*emult* F) $\mathbf{0}_I$ $\mathbf{1}_I$
and le : *ext-le* E F **and** a : $a \in \text{ecarrier } E$ **and** p : $\bigwedge k. p\ k \in \text{ecarrier } E$
shows *eval* F a p = *eval* E a p
 ⟨*proof*⟩

71.6 The base extension

The constants form an extension: a copy of R inside the indexed polynomial ring, carrying the ambient operations restricted to it. Besides supplying the upper bound Zorn needs for the *empty* chain, this is the check that *is-extension* is satisfiable at all — a condition as encoded as *root-closed* could easily be vacuous, and then everything proved from it would be too. Here *root-closed* holds

vacuously *by design*: a constant involves no indeterminate, so its hypothesis is never met.

definition *base-ext* :: 'a iext
where *base-ext* =
 (ipoly-const ' R,
 ($\lambda x y.$ if $x \in \text{ipoly-const ' R} \wedge y \in \text{ipoly-const ' R}$ then $x \oplus_I y$ else undefined),
 ($\lambda x y.$ if $x \in \text{ipoly-const ' R} \wedge y \in \text{ipoly-const ' R}$ then $x \otimes_I y$ else undefined))

lemma *base-ext-carrier* [simp]: *ecarrier base-ext* = *ipoly-const ' R*
 <proof>

lemma *base-ext-add*:
assumes $a \in R$ **and** $b \in R$
shows *eadd base-ext* (*ipoly-const a*) (*ipoly-const b*) = *ipoly-const* ($a + b$)
 <proof>

lemma *base-ext-mult*:
assumes $a \in R$ **and** $b \in R$
shows *emult base-ext* (*ipoly-const a*) (*ipoly-const b*) = *ipoly-const* ($a \cdot b$)
 <proof>

lemma *base-ext-restricted*:
op-restricted (*ecarrier base-ext*) (*eadd base-ext*)
op-restricted (*ecarrier base-ext*) (*emult base-ext*)
 <proof>

71.7 Unions of chains

A chain of extensions: a subset of *extensions* any two of whose members are comparable. Its union is again an extension, and is an upper bound — which is what Zorn's lemma asks for.

definition *ext-chain* :: 'a iext set $\Rightarrow$ bool
where *ext-chain* $C \iff C \subseteq \text{extensions} \wedge (\forall E \in C. \forall F \in C. \text{ext-le } E F \vee \text{ext-le } F E)$

lemma *ext-chainD-extension*: $\llbracket \text{ext-chain } C; E \in C \rrbracket \implies \text{is-extension } E$
 <proof>

lemma *ext-chainD-comparable*: $\llbracket \text{ext-chain } C; E \in C; F \in C \rrbracket \implies \text{ext-le } E F \vee \text{ext-le } F E$
 <proof>

The union carries, on each pair of elements, the operations of any member containing both; the chain condition makes the choice immaterial.

definition *ext-Union* :: 'a iext set $\Rightarrow$ 'a iext
where *ext-Union* $C =$
 ($\bigcup (\text{ecarrier ' C}),$
 ($\lambda x y.$ if $x \in \bigcup (\text{ecarrier ' C}) \wedge y \in \bigcup (\text{ecarrier ' C})$

$$\text{then eadd (SOME } E. E \in C \wedge x \in \text{ecarrier } E \wedge y \in \text{ecarrier } E) \text{ } x \text{ } y \text{ else}$$

$$\text{undefined}),$$

$$(\lambda x \text{ } y. \text{ if } x \in \bigcup (\text{ecarrier } 'C) \wedge y \in \bigcup (\text{ecarrier } 'C)$$

$$\text{then emult (SOME } E. E \in C \wedge x \in \text{ecarrier } E \wedge y \in \text{ecarrier } E) \text{ } x \text{ } y \text{ else}$$

$$\text{undefined}))$$

lemma *ext-Union-carrier* [simp]: $\text{ecarrier (ext-Union } C) = \bigcup (\text{ecarrier } 'C)$
 $\langle \text{proof} \rangle$

lemma *ext-Union-restricted*:
 $\text{op-restricted (ecarrier (ext-Union } C)) (\text{eadd (ext-Union } C))$
 $\text{op-restricted (ecarrier (ext-Union } C)) (\text{emult (ext-Union } C))$
 $\langle \text{proof} \rangle$

context
fixes $C :: 'a \text{ iext set}$
assumes *chain*: $\text{ext-chain } C$
begin

lemma *chain-field*: $E \in C \implies \text{Field (ecarrier } E) (\text{eadd } E) (\text{emult } E) \mathbf{0}_I \mathbf{1}_I$
 $\langle \text{proof} \rangle$

Two elements of the union lie in a common member: the larger of two members containing them.

lemma *chain-common2*:
assumes $x \in \text{ecarrier (ext-Union } C)$ **and** $y \in \text{ecarrier (ext-Union } C)$
shows $\exists E \in C. x \in \text{ecarrier } E \wedge y \in \text{ecarrier } E$
 $\langle \text{proof} \rangle$

lemma *chain-common3*:
assumes $x \in \text{ecarrier (ext-Union } C)$ **and** $y \in \text{ecarrier (ext-Union } C)$
and $z \in \text{ecarrier (ext-Union } C)$
shows $\exists E \in C. x \in \text{ecarrier } E \wedge y \in \text{ecarrier } E \wedge z \in \text{ecarrier } E$
 $\langle \text{proof} \rangle$

The union's operations agree with those of any member containing both arguments.

lemma *ext-Union-ops*:
assumes $E: E \in C$ **and** $x: x \in \text{ecarrier } E$ **and** $y: y \in \text{ecarrier } E$
shows $\text{eadd (ext-Union } C) \text{ } x \text{ } y = \text{eadd } E \text{ } x \text{ } y$
and $\text{emult (ext-Union } C) \text{ } x \text{ } y = \text{emult } E \text{ } x \text{ } y$
 $\langle \text{proof} \rangle$

lemma *ext-Union-upper-bound*: $E \in C \implies \text{ext-le } E (\text{ext-Union } C)$
 $\langle \text{proof} \rangle$

Every member's carrier contains $\mathbf{0}_I$ and $\mathbf{1}_I$, being an extension of R .

lemma *chain-zero-one-mem*:

assumes $E: E \in C$ **shows** $\mathbf{0}_I \in \text{ecarrier } E$ **and** $\mathbf{1}_I \in \text{ecarrier } E$
 $\langle \text{proof} \rangle$

The field laws of a single member, extracted through one interpretation so that the union's proof below can cite them without re-interpreting for every axiom.

lemma *chain-member*:

assumes $E: E \in C$

shows *chain-add-closed*: $\bigwedge x y. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E \rrbracket$
 $\implies \text{eadd } E x y \in \text{ecarrier } E$

and *chain-mult-closed*: $\bigwedge x y. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E \rrbracket$
 $\implies \text{emult } E x y \in \text{ecarrier } E$

and *chain-add-assoc*: $\bigwedge x y z. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E; z \in \text{ecarrier } E \rrbracket$
 $\implies \text{eadd } E (\text{eadd } E x y) z = \text{eadd } E x (\text{eadd } E y z)$

and *chain-mult-assoc*: $\bigwedge x y z. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E; z \in \text{ecarrier } E \rrbracket$
 $\implies \text{emult } E (\text{emult } E x y) z = \text{emult } E x (\text{emult } E y z)$

and *chain-add-zero*: $\bigwedge x. x \in \text{ecarrier } E \implies \text{eadd } E \mathbf{0}_I x = x$

and *chain-add-zero'*: $\bigwedge x. x \in \text{ecarrier } E \implies \text{eadd } E x \mathbf{0}_I = x$

and *chain-mult-one*: $\bigwedge x. x \in \text{ecarrier } E \implies \text{emult } E \mathbf{1}_I x = x$

and *chain-mult-one'*: $\bigwedge x. x \in \text{ecarrier } E \implies \text{emult } E x \mathbf{1}_I = x$

and *chain-add-comm*: $\bigwedge x y. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E \rrbracket$
 $\implies \text{eadd } E x y = \text{eadd } E y x$

and *chain-mult-comm*: $\bigwedge x y. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E \rrbracket$
 $\implies \text{emult } E x y = \text{emult } E y x$

and *chain-distrib*: $\bigwedge x y z. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E; z \in \text{ecarrier } E \rrbracket$
 $\implies \text{emult } E x (\text{eadd } E y z) = \text{eadd } E (\text{emult } E x y) (\text{emult } E x z)$

and *chain-distrib'*: $\bigwedge x y z. \llbracket x \in \text{ecarrier } E; y \in \text{ecarrier } E; z \in \text{ecarrier } E \rrbracket$
 $\implies \text{emult } E (\text{eadd } E y z) x = \text{eadd } E (\text{emult } E y x) (\text{emult } E z x)$

$E z x$

and *chain-add-inv*: $\bigwedge x. x \in \text{ecarrier } E$
 $\implies \exists y \in \text{ecarrier } E. \text{eadd } E x y = \mathbf{0}_I \wedge \text{eadd } E y x = \mathbf{0}_I$

and *chain-mult-inv*: $\bigwedge x. \llbracket x \in \text{ecarrier } E; x \neq \mathbf{0}_I \rrbracket$
 $\implies \exists y \in \text{ecarrier } E. \text{emult } E x y = \mathbf{1}_I \wedge \text{emult } E y x = \mathbf{1}_I$

$\langle \text{proof} \rangle$

The union of a nonempty chain is a field. Each axiom involves at most three elements, which a common member accommodates; there the axiom holds, and the union's operations agree with that member's.

lemma *ext-Union-field*:

assumes $ne: C \neq \{\}$ **and** $nz: \mathbf{1} \neq \mathbf{0}$

shows $\text{Field } (\text{ecarrier } (\text{ext-Union } C)) (\text{eadd } (\text{ext-Union } C)) (\text{emult } (\text{ext-Union } C))$
 $\mathbf{0}_I \mathbf{1}_I$

$\langle \text{proof} \rangle$

71.8 Evaluation is the same in a member and in the union

The remaining extension condition, *root-closed*, asserts that an indeterminate is a *root*, which is an equation about evaluation. Evaluation does not transfer between a member and the union for free: *commutative-monoid.fincomp* depends

on the carrier as well as the operation (through *commutative-monoid.M-ify*), so *fincomp-transfer* is what makes the two agree, given that every value involved stays inside the member.

lemma *chain-Ring*:

assumes $E: E \in C$ **shows** $\text{Ring } (\text{ecarrier } E) (\text{eadd } E) (\text{emult } E) \mathbf{0}_I \mathbf{1}_I$
 $\langle \text{proof} \rangle$

lemma *ext-Union-Ring*:

assumes $ne: C \neq \{\}$ **and** $nz: \mathbf{1} \neq \mathbf{0}$
shows $\text{Ring } (\text{ecarrier } (\text{ext-Union } C)) (\text{eadd } (\text{ext-Union } C)) (\text{emult } (\text{ext-Union } C))$
 $\mathbf{0}_I \mathbf{1}_I$
 $\langle \text{proof} \rangle$

The chain case is now just the ordering case at *ext-Union-upper-bound*.

lemma *rpow-transfer*:

assumes $ne: C \neq \{\}$ **and** $nz: \mathbf{1} \neq \mathbf{0}$ **and** $E: E \in C$ **and** $a: a \in \text{ecarrier } E$
shows $\text{Ring.rpow } (\text{emult } (\text{ext-Union } C)) \mathbf{1}_I a n = \text{Ring.rpow } (\text{emult } E) \mathbf{1}_I a n$
 $\langle \text{proof} \rangle$

lemma *ext-Union-eval*:

assumes $ne: C \neq \{\}$ **and** $nz: \mathbf{1} \neq \mathbf{0}$ **and** $E: E \in C$
and $a: a \in \text{ecarrier } E$ **and** $p: \bigwedge k. p \ k \in \text{ecarrier } E$
shows $\text{eval } (\text{ext-Union } C) a p = \text{eval } E a p$
 $\langle \text{proof} \rangle$

71.9 The union of a nonempty chain is an extension

lemma *ext-Union-is-extension*:

assumes $ne: C \neq \{\}$ **and** $nz: \mathbf{1} \neq \mathbf{0}$
shows $\text{is-extension } (\text{ext-Union } C)$
 $\langle \text{proof} \rangle$

end

end

72 Transporting a Field Structure along an Injection

context *Field*

begin

A field structure transports along a map injective on the carrier, given operations on the image that make the map a homomorphism there. Only the behaviour *on the image* is constrained, which is what makes the restricted operations of an extension free of charge: no separate lemma is needed to see that restricting a field's operations to its own carrier leaves a field. The same lemma will serve when a root is adjoined, since that too realises a field as the image of an injection.

lemma *field-image-transport*:
fixes $h :: 'a \Rightarrow 'b$ **and** $\text{hadd hmult} :: 'b \Rightarrow 'b \Rightarrow 'b$
assumes $\text{inj}: \text{inj-on } h \ R$
and $\text{ha}: \bigwedge a \ b. \llbracket a \in R; b \in R \rrbracket \Longrightarrow \text{hadd } (h \ a) \ (h \ b) = h \ (a + b)$
and $\text{hm}: \bigwedge a \ b. \llbracket a \in R; b \in R \rrbracket \Longrightarrow \text{hmult } (h \ a) \ (h \ b) = h \ (a \cdot b)$
shows $\text{Field } (h \ ` \ R) \ \text{hadd hmult } (h \ 0) \ (h \ 1)$
 $\langle \text{proof} \rangle$

72.1 The base extension is an extension

theorem *is-extension-base-ext*: $\text{is-extension base-ext}$
 $\langle \text{proof} \rangle$

corollary *extensions-non-empty*: $\text{extensions} \neq \{\}$
 $\langle \text{proof} \rangle$

72.2 A maximal extension exists

Zorn's lemma applies: the extensions form a set, partially ordered by *ext-le*, in which every chain has an upper bound — the union of a nonempty chain, and *base-ext* for the empty one. Maximality is what will force every polynomial over R to split, since an unsplit polynomial leaves an index free at which a root could be adjoined.

theorem *exists-maximal-extension*:
 $\exists M \in \text{extensions}. \forall L \in \text{extensions}. \text{ext-le } M \ L \longrightarrow L = M$
 $\langle \text{proof} \rangle$

72.3 A fresh index always exists

Towards *exists-root*: for any nonzero polynomial P over R there is an index i that no element of a given extension involves. This is the step that makes room for a new root, and it is where the choice of indexed polynomials as the ambient type pays off — no cardinality argument is needed, only the observation that an extension *mentioning* every index would contain a root of P for each of them, and a nonzero polynomial over a field has only finitely many roots.

lemma *exists-fresh-index*:
assumes M : *is-extension* M **and** P : $P \in \text{poly-carrier}$ **and** Pnz : $P \neq 0_P$
shows $\exists i. \forall \mathcal{P} \in \text{ecarrier } M. \text{index-free } \mathcal{P} \ (P, i)$
 $\langle \text{proof} \rangle$

end

72.4 Transporting a field onto its image, with restricted operations

An injective image of a field carries a field structure, and the operations obtained by *transport* — undo the injection, operate, redo it — are automatically the

restricted ones an extension requires. Nothing is assumed about how h interacts with the ambient operations, which is the point: an extension's addition need not be the ambient addition, and *Ring.is-extension* does not require it to be. Choosing the operations by transport sidesteps the question instead of forcing a stronger definition.

lemma *field-transport-restricted*:

```

fixes  $A :: 'b \text{ set}$  and  $\text{add mult} :: 'b \Rightarrow 'b \Rightarrow 'b$  and  $z u :: 'b$ 
and  $h :: 'b \Rightarrow 'a \text{ ipoly}$ 
assumes  $F$ : Field  $A$   $\text{add mult } z u$  and  $\text{inj}$ : inj-on  $h A$ 
defines  $K \equiv h \text{ ` } A$ 
and  $\text{hadd} \equiv (\lambda x y. \text{if } x \in h \text{ ` } A \wedge y \in h \text{ ` } A$ 
   $\text{then } h (\text{add } (\text{inv-into } A h x) (\text{inv-into } A h y)) \text{ else undefined})$ 
and  $\text{hmult} \equiv (\lambda x y. \text{if } x \in h \text{ ` } A \wedge y \in h \text{ ` } A$ 
   $\text{then } h (\text{mult } (\text{inv-into } A h x) (\text{inv-into } A h y)) \text{ else undefined})$ 
shows field-transport-field: Field  $K$   $\text{hadd hmult } (h z) (h u)$ 
and field-transport-add-restricted: op-restricted  $K$   $\text{hadd}$ 
and field-transport-mult-restricted: op-restricted  $K$   $\text{hmult}$ 
 $\langle \text{proof} \rangle$ 

```

73 Working Inside a Fixed Extension

Adjoining a root needs the polynomial machinery of the extension's own coefficient field — *pmod*, *poly-carrier*, *degree* at *ecarrier* M . Fixing M in a locale and discharging *Field* for it once, by *Ring.is-extension-field*, makes all of it available under the M prefix throughout, *including inside definitions*. **interpret** would not serve, being a proof-body command, but **sublocale** is its declaration-level counterpart, so no definition here has to spell out the five locale parameters.

```

locale field-extension = Field +
fixes  $M :: 'a \text{ iext}$ 
assumes is-ext: is-extension  $M$ 

```

```

sublocale field-extension  $\subseteq M$ : Field ecarrier  $M$  eadd  $M$  emult  $M$  0I 1I
 $\langle \text{proof} \rangle$ 

```

```

context field-extension
begin

```

The realisation map: reduce a polynomial over M modulo Q , then read the remainder off at the fresh indeterminate, using the *ambient* operations. Both prefixes now read naturally.

```

definition realise ::  $'a \text{ root-index} \Rightarrow (\text{nat} \Rightarrow 'a \text{ ipoly}) \Rightarrow (\text{nat} \Rightarrow 'a \text{ ipoly}) \Rightarrow 'a \text{ ipoly}$ 
where  $\text{realise } j Q S = \text{ieval } (\text{ivar } j) (M.\text{pmod } S Q)$ 

```

A polynomial over M is also one over the ambient ring, its carrier being contained there. Note that the two rings share their zero, so they also share *Ring.degree* and *Ring.poly-zero*: no translation of degrees is needed anywhere below.

lemma *coeff-ipoly-closed*: $S \in M.\text{poly-carrier} \implies S\ k \in \text{ipoly-carrier}$
 ⟨proof⟩

lemma *low-poly-ambient*:

assumes $S: S \in M.\text{low-poly}\ d$ **shows** $S \in \text{Ring.poly-carrier ipoly-carrier } \mathbf{0}_I$
 ⟨proof⟩

On the remainders the realisation is a sum over the *fixed* range $\{..d\}$ rather than over each polynomial's own degree. A common range is what *ieval-ivar-inj* needs, and enlarging the range past the degree is harmless.

lemma *ieval-low-fincomp*:

assumes $S: S \in M.\text{low-poly}\ d$
shows $\text{ieval } (\text{ivar } j)\ S$
 $= \text{commutative-monoid.fincomp ipoly-carrier } (\oplus_I)\ \mathbf{0}_I$
 $(\lambda k. S\ k \otimes_I \text{Ring.rpow } (\otimes_I)\ \mathbf{1}_I\ (\text{ivar } j)\ k)\ \{..d\}$
 ⟨proof⟩

The realisation is injective on the remainders. This is the point of a *fresh* index: every coefficient lies in M and so is index-free at j , which is exactly the hypothesis under which *ieval-ivar-inj* reads the coefficients back off. Above the range the two remainders agree trivially, both coefficients being zero.

lemma *ieval-inj-on-low*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in \text{ecarrier } M \implies \text{index-free } \mathcal{P}\ j$
shows $\text{inj-on } (\text{ieval } (\text{ivar } j))\ (M.\text{low-poly}\ d)$
 ⟨proof⟩

The extension obtained by adjoining a root. The carrier is the image of the remainders modulo Q under evaluation at the fresh indeterminate; the operations are the transported ones. Reduction modulo Q has to live in the multiplication, since the realisation of Q itself is not zero in the ambient ring — which is exactly why an extension carries its own operations rather than inheriting the ambient ones.

definition *adjoin-carrier* :: $'a\ \text{root-index} \Rightarrow (\text{nat} \Rightarrow 'a\ \text{ipoly}) \Rightarrow 'a\ \text{ipoly set}$
where $\text{adjoin-carrier } j\ Q = \text{ieval } (\text{ivar } j)\ 'M.\text{low-poly } (M.\text{degree } Q)$

definition *adjoin* :: $'a\ \text{root-index} \Rightarrow (\text{nat} \Rightarrow 'a\ \text{ipoly}) \Rightarrow 'a\ \text{iext}$

where $\text{adjoin } j\ Q =$
 $(\text{adjoin-carrier } j\ Q,$
 $(\lambda x\ y. \text{if } x \in \text{adjoin-carrier } j\ Q \wedge y \in \text{adjoin-carrier } j\ Q$
 $\text{then } \text{ieval } (\text{ivar } j)$
 $\quad (M.\text{poly-add } (\text{inv-into } (M.\text{low-poly } (M.\text{degree } Q))\ (\text{ieval } (\text{ivar } j))\ x)$
 $\quad (\text{inv-into } (M.\text{low-poly } (M.\text{degree } Q))\ (\text{ieval } (\text{ivar } j))\ y))$
 $\text{else undefined}),$
 $(\lambda x\ y. \text{if } x \in \text{adjoin-carrier } j\ Q \wedge y \in \text{adjoin-carrier } j\ Q$
 $\text{then } \text{ieval } (\text{ivar } j)$
 $\quad (M.\text{pmod } (M.\text{poly-mult}$
 $\quad (\text{inv-into } (M.\text{low-poly } (M.\text{degree } Q))\ (\text{ieval } (\text{ivar } j))\ x)$
 $\quad (\text{inv-into } (M.\text{low-poly } (M.\text{degree } Q))\ (\text{ieval } (\text{ivar } j))\ y))\ Q)$

else undefined))

lemma *adjoin-carrier-eq* [simp]: $\text{ecarrier } (\text{adjoin } j \ Q) = \text{adjoin-carrier } j \ Q$
 ⟨proof⟩

The field structure and the restrictedness of both operations come together, by transporting *Field.remainder-field* along the realisation, which *ieval-inj-on-low* shows to be injective on the remainders. The zero and the unit land on $\mathbf{0}_I$ and $\mathbf{1}_I$ because evaluation sends the zero and unit polynomials there.

The realisation sends the zero, the unit and the variable of the polynomial ring over M to $\mathbf{0}_I$, $\mathbf{1}_I$ and the fresh indeterminate. The first two are what make the transported field structure have the zero and unit that *is-extension* demands; the third is what exhibits *ivar* j as an element of the adjoined carrier.

lemma *ieval-poly-zero*: $\text{ieval } (\text{ivar } j) \ M.\text{poly-zero} = \mathbf{0}_I$
 ⟨proof⟩

lemma *ieval-poly-one*: $\text{ieval } (\text{ivar } j) \ M.\text{poly-one} = \mathbf{1}_I$
 ⟨proof⟩

lemma *ieval-poly-var*: $\text{ieval } (\text{ivar } j) \ M.\text{var} = \text{ivar } j$
 ⟨proof⟩

Nontriviality of the ambient indexed ring, taken from R rather than from M : a statement about $\mathbf{1}_I$ and $\mathbf{0}_I$ alone is generalised over the index type and so cannot be derived from a hypothesis where that type is pinned.

lemma *ipoly-one-neq-zero*: $(\mathbf{1}_I :: 'a \text{ root-index multiset} \Rightarrow 'a) \neq \mathbf{0}_I$
 ⟨proof⟩

lemma *var-low*:
 assumes $d2: 2 \leq M.\text{degree } Q$ shows $M.\text{var} \in M.\text{low-poly } (M.\text{degree } Q)$
 ⟨proof⟩

The constants of M sit inside the adjoined carrier, realised by the constant polynomials, and the transported operations restrict to M 's own there. So M lies below the adjoined extension in the ordering — which is what carries M 's root witnesses upwards, by *Ring.ext-le-eval*.

lemma *M-Ring*: $\text{Ring } (\text{ecarrier } M) \ (\text{eadd } M) \ (\text{emult } M) \ \mathbf{0}_I \ \mathbf{1}_I$
 ⟨proof⟩

lemma *adjoin-rep*:
 assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in \text{ecarrier } M \implies \text{index-free } \mathcal{P} \ j$
 and $S: S \in M.\text{low-poly } (M.\text{degree } Q)$
 shows $\text{inv-into } (M.\text{low-poly } (M.\text{degree } Q)) \ (\text{ieval } (\text{ivar } j)) \ (\text{ieval } (\text{ivar } j) \ S) = S$
 ⟨proof⟩

lemma *ieval-poly-const*:
 assumes $y: y \in \text{ipoly-carrier}$ shows $\text{ieval } (\text{ivar } j) \ (M.\text{poly-const } y) = y$

$\langle proof \rangle$

lemma *const-low*:

assumes *irr*: $M.poly\text{-}irreducible\ Q$ **and** $y: y \in ecarrier\ M$
shows $M.poly\text{-}const\ y \in M.low\text{-}poly\ (M.degree\ Q)$

$\langle proof \rangle$

lemma *const-rep*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in ecarrier\ M \implies index\text{-}free\ \mathcal{P}\ j$
and *irr*: $M.poly\text{-}irreducible\ Q$ **and** $y: y \in ecarrier\ M$
shows $inv\text{-}into\ (M.low\text{-}poly\ (M.degree\ Q))\ (ieval\ (ivar\ j))\ y = M.poly\text{-}const\ y$

$\langle proof \rangle$

lemma *ecarrier-subset-adjoin*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in ecarrier\ M \implies index\text{-}free\ \mathcal{P}\ j$
and *irr*: $M.poly\text{-}irreducible\ Q$
shows $ecarrier\ M \subseteq adjoin\text{-}carrier\ j\ Q$

$\langle proof \rangle$

lemma *ext-le-adjoin*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in ecarrier\ M \implies index\text{-}free\ \mathcal{P}\ j$
and *irr*: $M.poly\text{-}irreducible\ Q$
shows $ext\text{-}le\ M\ (adjoin\ j\ Q)$

$\langle proof \rangle$

lemma *adjoin-field*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in ecarrier\ M \implies index\text{-}free\ \mathcal{P}\ j$
and *irr*: $M.poly\text{-}irreducible\ Q$
shows $Field\ (ecarrier\ (adjoin\ j\ Q))\ (eadd\ (adjoin\ j\ Q))\ (emult\ (adjoin\ j\ Q))$
 $(ieval\ (ivar\ j)\ M.poly\text{-}zero)\ (ieval\ (ivar\ j)\ M.poly\text{-}one)$
and $op\text{-}restricted\ (ecarrier\ (adjoin\ j\ Q))\ (eadd\ (adjoin\ j\ Q))$
and $op\text{-}restricted\ (ecarrier\ (adjoin\ j\ Q))\ (emult\ (adjoin\ j\ Q))$

$\langle proof \rangle$

The adjoined field, with the zero and unit *is-extension* demands.

lemma *adjoin-field-zero-one*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in ecarrier\ M \implies index\text{-}free\ \mathcal{P}\ j$
and *irr*: $M.poly\text{-}irreducible\ Q$
shows $Field\ (ecarrier\ (adjoin\ j\ Q))\ (eadd\ (adjoin\ j\ Q))\ (emult\ (adjoin\ j\ Q))$
 $\mathbf{0}_I\ \mathbf{1}_I$

$\langle proof \rangle$

lemma *adjoin-Ring*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in ecarrier\ M \implies index\text{-}free\ \mathcal{P}\ j$
and *irr*: $M.poly\text{-}irreducible\ Q$
shows $Ring\ (ecarrier\ (adjoin\ j\ Q))\ (eadd\ (adjoin\ j\ Q))\ (emult\ (adjoin\ j\ Q))$
 $\mathbf{0}_I\ \mathbf{1}_I$

$\langle proof \rangle$

lemma *lift-poly-M-carrier*:

assumes $P: P \in \text{poly-carrier}$ **shows** $\text{lift-poly } P \in M.\text{poly-carrier}$
 $\langle \text{proof} \rangle$

Root-closedness at the adjoined index itself. The fresh indeterminate is the realisation of the variable, so it lies in the carrier once the modulus has degree at least two. And it is a root: *eval-transport* moves the evaluation into the field of remainders, where *eval-remainder-at-var* identifies it with the reduction of the polynomial — which vanishes precisely because the modulus divides it.

lemma *root-closed-adjoin-self:*

assumes *fresh:* $\bigwedge \mathcal{P}. \mathcal{P} \in \text{ecarrier } M \implies \text{index-free } \mathcal{P} \ j$
and *irr:* $M.\text{poly-irreducible } Q$
and *dvd:* $M.\text{poly-dvd } Q \ (\text{lift-poly } (\text{fst } j))$
and *Pj:* $\text{fst } j \in \text{poly-carrier}$
and *d2:* $2 \leq M.\text{degree } Q$
shows $\text{ivar } j \in \text{ecarrier } (\text{adjoin } j \ Q)$
 $\wedge \text{eeval } (\text{adjoin } j \ Q) \ (\text{ivar } j) \ (\text{lift-poly } (\text{fst } j)) = \mathbf{0}_I$
 $\langle \text{proof} \rangle$

Root-closedness at an index other than the adjoined one. An element of the adjoined carrier that involves such an index must have a coefficient involving it — removing copies of the adjoined index cannot remove a different one — and that coefficient lies in M , where root-closedness applies. *Ring.ext-le-eval* then carries the root upwards.

lemma *root-closed-adjoin-other:*

assumes *fresh:* $\bigwedge \mathcal{P}. \mathcal{P} \in \text{ecarrier } M \implies \text{index-free } \mathcal{P} \ j$
and *irr:* $M.\text{poly-irreducible } Q$
and *S:* $S \in M.\text{low-poly } (M.\text{degree } Q)$
and *P':* $P' \in \text{poly-carrier}$ **and** *ne:* $(P', i') \neq j$
and *nf:* $\neg \text{index-free } (\text{ieval } (\text{ivar } j) \ S) \ (P', i')$
shows $\text{ivar } (P', i') \in \text{ecarrier } (\text{adjoin } j \ Q)$
 $\wedge \text{eeval } (\text{adjoin } j \ Q) \ (\text{ivar } (P', i')) \ (\text{lift-poly } P') = \mathbf{0}_I$
 $\langle \text{proof} \rangle$

The adjoined extension is root-closed. The two cases above, joined by the observation that the adjoined index can only be involved at all when the modulus is nonlinear: were its degree at most one the remainders would be constants, so every element of the carrier would lie in M and be index-free at the fresh index by freshness.

lemma *root-closed-adjoin:*

assumes *fresh:* $\bigwedge \mathcal{P}. \mathcal{P} \in \text{ecarrier } M \implies \text{index-free } \mathcal{P} \ j$
and *irr:* $M.\text{poly-irreducible } Q$
and *dvd:* $M.\text{poly-dvd } Q \ (\text{lift-poly } (\text{fst } j))$
and *Pj:* $\text{fst } j \in \text{poly-carrier}$
shows *root-closed* $(\text{adjoin } j \ Q)$
 $\langle \text{proof} \rangle$

The adjoined triple is an extension. Everything is now in place: the field structure and the restrictedness of both operations come from the transport, root-closedness from the two cases above, and the conditions on the constants from

the fact that M lies below, its own agreement with R on constants carrying over.

theorem *is-extension-adjoin*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in \text{ecarrier } M \implies \text{index-free } \mathcal{P} \ j$

and *irr*: $M.\text{poly-irreducible } Q$

and *dvd*: $M.\text{poly-dvd } Q \ (\text{lift-poly } (\text{fst } j))$

and *Pj*: $\text{fst } j \in \text{poly-carrier}$

shows *is-extension* ($\text{adjoin } j \ Q$)

<proof>

And it is *strictly* larger: the fresh indeterminate involves its own index, whereas every element of M is index-free there.

lemma *adjoin-strictly-larger*:

assumes *fresh*: $\bigwedge \mathcal{P}. \mathcal{P} \in \text{ecarrier } M \implies \text{index-free } \mathcal{P} \ j$

and *irr*: $M.\text{poly-irreducible } Q$ **and** *d2*: $2 \leq M.\text{degree } Q$

shows $\text{adjoin } j \ Q \neq M$

<proof>

73.1 Every polynomial over the base field splits in a maximal extension

The point of maximality. Factor the lifted polynomial into irreducibles over M . A factor of degree two or more would let a root of *it* be adjoined at a fresh index — giving a strictly larger extension, which maximality forbids. So every factor is linear, and a product of linear factors splits.

Note where each earlier piece enters: *exists-fresh-index* supplies the index, *is-extension-adjoin* and *ext-le-adjoin* make the enlargement an extension above M , *adjoin-strictly-larger* contradicts maximality, and *Field.degree-one-splits* with *Field.splits-poly-prod* assembles the conclusion.

theorem *exists-root*:

assumes *max*: $\bigwedge L. \llbracket \text{is-extension } L; \text{ext-le } M \ L \rrbracket \implies L = M$

and *P*: $P \in \text{poly-carrier}$

shows $M.\text{splits } (\text{lift-poly } P)$

<proof>

end

74 An Extension in which Every Base Polynomial Splits

context *Field*

begin

Zorn gave a maximal extension; maximality makes every polynomial over R split in it. Putting the two together is the whole of this step, exactly as it is in HOL-Algebra.

```

theorem exists-extension-with-roots:
  obtains  $M$  where is-extension  $M$ 
    and  $\bigwedge P. P \in \text{poly-carrier}$ 
       $\implies \text{Field.splits } (\text{ecarrier } M) (\text{eadd } M) (\text{emult } M) \mathbf{0}_I \mathbf{1}_I (\text{lift-poly } P)$ 
   $\langle \text{proof} \rangle$ 

end

end

```

75 Linking the carrier-set polynomial ring to Isabelle's polynomial type

```

theory Poly-Typeclass
imports Poly-Ring Ring-Theory HOL-Computational-Algebra.Polynomial
begin

```

Poly-Ring represents a polynomial as a finite-support function $\text{nat} \Rightarrow 'a$ with coefficients in a carrier R , which is exactly the representing set of HOL's *'a poly* once the carrier is the whole type. So for a type-class field the two developments describe the same objects, and the dictionary between them is little more than *Abs-poly*: this theory writes it out, so that the ring-theoretic results proved for *poly-carrier* — irreducibility, division, unique factorisation — become available to the extension and Galois theory, which is phrased throughout in terms of *'a poly*.

75.1 Finite composites are sums and products

The carrier-set *commutative-monoid.fincomp* at the whole type is HOL's *sum* or *prod*; the guard *commutative-monoid.M-ify* in its definition is vacuous there.

```

lemma fincomp-TC-sum:
  commutative-monoid.fincomp (UNIV :: 'a :: comm-monoid-add set) (+) 0  $f$   $A$  =
    sum  $f$   $A$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma fincomp-TC-prod:
  commutative-monoid.fincomp (UNIV :: 'a :: comm-monoid-mult set) (*) 1  $f$   $A$  =
    prod  $f$   $A$ 
   $\langle \text{proof} \rangle$ 

```

75.2 The two representations of a polynomial

A finite-support function read as a polynomial. Off the finite-support functions the value is arbitrary; 0 is chosen so that the map is total.

```

definition poly-of-fun :: ( $\text{nat} \Rightarrow 'a :: \text{zero}$ )  $\Rightarrow$  'a poly
  where poly-of-fun  $p \equiv (\text{if finite } \{i. p\ i \neq 0\} \text{ then } \text{Abs-poly } p \text{ else } 0)$ 

```

lemma *coeff-poly-of-fun* [simp]:
 $\text{finite } \{i. p \ i \neq 0\} \implies \text{Polynomial.coeff } (\text{poly-of-fun } p) = p$
 ⟨proof⟩

lemma *finite-support-coeff* [simp, intro]: $\text{finite } \{i. \text{Polynomial.coeff } p \ i \neq 0\}$
 ⟨proof⟩

lemma *poly-of-fun-coeff* [simp]: $\text{poly-of-fun } (\text{Polynomial.coeff } p) = p$
 ⟨proof⟩

lemma *inj-on-poly-of-fun*: $\text{inj-on poly-of-fun } \{p. \text{finite } \{i. p \ i \neq 0\}\}$
 ⟨proof⟩

75.3 The dictionary at a type-class field

Fixing the carrier to be the whole type turns every constant of *Poly-Ring* into an operation on finite-support functions over a type-class field; the locale below exists only to make those constants available unqualified.

locale *poly-TC* = *Field UNIV* :: 'a :: field set (+) (*) 0 1

interpretation *poly-TC*
 ⟨proof⟩

context *poly-TC*
begin

Membership of the carrier is finite support alone: every coefficient lies in the carrier because the carrier is everything.

lemma *poly-carrier-TC*: $\text{poly-carrier} = \{p :: \text{nat} \Rightarrow 'a. \text{finite } \{i. p \ i \neq 0\}\}$
 ⟨proof⟩

lemma *poly-carrier-iff*: $p \in \text{poly-carrier} \longleftrightarrow \text{finite } \{i. p \ i \neq 0\}$
 ⟨proof⟩

lemma *coeff-in-poly-carrier* [simp, intro]: $\text{Polynomial.coeff } q \in \text{poly-carrier}$
 ⟨proof⟩

lemma *coeff-poly-of-fun-carrier* [simp]:
 $p \in \text{poly-carrier} \implies \text{Polynomial.coeff } (\text{poly-of-fun } p) = p$
 ⟨proof⟩

The additive inverse supplied by the group structure is HOL's, though the two are distinct constants and print alike; without this the coefficientwise proofs below cannot close.

lemma *additive-inverse-TC* [simp]: $\text{additive.inverse } a = \text{uminus } a$
 ⟨proof⟩

On the carrier the dictionary is injective, so an identity between carrier-set polynomials may be checked in *'a poly* and conversely.

lemma *poly-of-fun-eq-iff* [simp]:

$\llbracket p \in \text{poly-carrier}; q \in \text{poly-carrier} \rrbracket \implies \text{poly-of-fun } p = \text{poly-of-fun } q \longleftrightarrow p = q$
 $\langle \text{proof} \rangle$

75.4 The ring operations correspond

lemma *poly-of-fun-zero* [simp]: $\text{poly-of-fun } \text{poly-zero} = 0$

$\langle \text{proof} \rangle$

lemma *poly-of-fun-one* [simp]: $\text{poly-of-fun } \text{poly-one} = 1$

$\langle \text{proof} \rangle$

lemma *poly-of-fun-add* [simp]:

assumes $p \in \text{poly-carrier}$ **and** $q \in \text{poly-carrier}$

shows $\text{poly-of-fun } (\text{poly-add } p \ q) = \text{poly-of-fun } p + \text{poly-of-fun } q$

$\langle \text{proof} \rangle$

lemma *poly-of-fun-neg* [simp]:

assumes $p \in \text{poly-carrier}$

shows $\text{poly-of-fun } (\text{poly-neg } p) = - \text{poly-of-fun } p$

$\langle \text{proof} \rangle$

The convolution defining *poly-mult* is literally HOL's *coeff-mult*, once the finite composite is read as a sum.

lemma *poly-of-fun-mult* [simp]:

assumes $p \in \text{poly-carrier}$ **and** $q \in \text{poly-carrier}$

shows $\text{poly-of-fun } (\text{poly-mult } p \ q) = \text{poly-of-fun } p * \text{poly-of-fun } q$

$\langle \text{proof} \rangle$

75.5 Constants, the indeterminate, degree and evaluation

lemma *poly-of-fun-const* [simp]: $\text{poly-of-fun } (\text{poly-const } c) = [:c:]$

$\langle \text{proof} \rangle$

lemma *poly-of-fun-monom* [simp]: $\text{poly-of-fun } (\text{monom } c \ k) = \text{Polynomial.monom } c$

k

$\langle \text{proof} \rangle$

lemma *poly-of-fun-var* [simp]: $\text{poly-of-fun } \text{var} = [:0, 1:]$

$\langle \text{proof} \rangle$

Both notions of degree are pinned by the same two inequalities: no coefficient above the degree, and a nonzero coefficient at it.

lemma *degree-poly-of-fun* [simp]:

assumes $p \in \text{poly-carrier}$

shows $\text{Polynomial.degree } (\text{poly-of-fun } p) = \text{degree } p$

$\langle \text{proof} \rangle$

Powers in the multiplicative monoid of the type are ordinary powers.

lemma *rpow-TC* [simp]: $\text{rpow } a \ n = a \wedge n$
 ⟨proof⟩

Evaluation matches *poly-altdef*, the coefficientwise form of *poly*.

lemma *poly-poly-of-fun* [simp]:
 assumes $p \in \text{poly-carrier}$
 shows $\text{Polynomial.poly } (\text{poly-of-fun } p) \ x = \text{eval } x \ p$
 ⟨proof⟩

75.6 Splitting into linear factors

lemma *poly-of-fun-root-factor* [simp]: $\text{poly-of-fun } (\text{root-factor } a) = [\text{:uminus } a, 1:]$
 ⟨proof⟩

lemma *poly-of-fun-poly-prod*:
 assumes $\bigwedge f. f \in \text{set } fs \implies f \in \text{poly-carrier}$
 shows $\text{poly-of-fun } (\text{poly-prod } fs) = (\prod f \leftarrow fs. \text{poly-of-fun } f)$
 ⟨proof⟩

The witnessing product, translated once for both directions of the theorem below.

lemma *poly-of-fun-split-form*:
 $\text{poly-of-fun } (\text{poly-mult } (\text{poly-const } c) (\text{poly-prod } (\text{List.map } \text{root-factor } as)))$
 $= [\text{:c:}] * (\prod a \leftarrow as. [\text{:uminus } a, 1:])$
 ⟨proof⟩

lemma *poly-mult-const-prod-carrier*: $\text{poly-mult } (\text{poly-const } c) (\text{poly-prod } (\text{List.map } \text{root-factor } as)) \in \text{poly-carrier}$
 ⟨proof⟩

splits says exactly that the corresponding polynomial of *'a poly* is a constant times a product of linear factors. Both directions are the dictionary applied to the defining equation, the reverse using injectivity on the carrier.

theorem *splits-iff-linear-factors*:
 assumes $p: p \in \text{poly-carrier}$
 shows $\text{splits } p \longleftrightarrow (\exists c \ as. \text{poly-of-fun } p = [\text{:c:}] * (\prod a \leftarrow as. [\text{:uminus } a, 1:]))$
 ⟨proof⟩

A splitting polynomial of positive degree has a root: the factor list cannot be empty.

theorem *splits-imp-root*:
 assumes $p: p \in \text{poly-carrier}$ and $sp: \text{splits } p$ and $d: 0 < \text{degree } p$
 obtains a where $\text{eval } a \ p = 0$
 ⟨proof⟩

75.7 The dictionary read backwards

Every polynomial of *'a poly* is the reading of its own coefficient function, so each correspondence above serves in the other direction too.

lemma *eval-coeff [simp]*: *eval x (Polynomial.coeff q) = Polynomial.poly q x*
<proof>

lemma *splits-coeff-iff*:
*splits (Polynomial.coeff q) $\longleftrightarrow (\exists c \text{ as. } q = [:c:] * (\prod a \leftarrow \text{as. } [:uminus a, 1:])))$*
<proof>

corollary *splits-coeff-imp-root*:
assumes *splits (Polynomial.coeff q)* **and** $0 < \text{Polynomial.degree } q$
obtains *a where Polynomial.poly q a = 0*
<proof>

end

Outside the locale the carrier-set degree must be named in full, its only parameter being the zero of the ring; that also makes it usable as a simplification rule on ordinary polynomials.

lemma *degree-coeff [simp]*:
fixes $q :: 'a :: \text{field poly}$
shows $\text{Ring.degree } 0 \text{ (Polynomial.coeff } q) = \text{Polynomial.degree } q$
<proof>

end

76 Degree of a simple algebraic extension

theory *Extension-Degree*
imports *Simple-Extension Extension-Vector-Space*
begin

This theory begins the *field side* of phase R2. For an element *a* algebraic over a subfield *K*, it turns the *predicate is-minpoly* into a genuine *function minpoly K a* (the minimal polynomial is unique) and defines the degree of the simple extension *K a* over *K* as *degree (minpoly K a)*.

76.1 Uniqueness of the minimal polynomial

context *Subfield*
begin

The minimal polynomial is unique: two minimal polynomials of *a* over *K* divide each other (each is a *K*-annihilator of *a*, so a multiple of the other by *minpoly-dvd*), and being monic they must be equal.

lemma *minpoly-unique*:
assumes $m1$: *is-minpoly* K a $m1$ **and** $m2$: *is-minpoly* K a $m2$
shows $m1 = m2$
 $\langle proof \rangle$

end

76.2 The minimal polynomial as a function

The minimal polynomial of a over K . For a algebraic over K this is the unique monic polynomial of least degree over K vanishing at a ; otherwise it is an unspecified default.

definition *minpoly* :: $'a :: \text{field set} \Rightarrow 'a \Rightarrow 'a \text{ poly}$
where *minpoly* K $a = (\text{SOME } m. \text{is-minpoly } K \text{ } a \text{ } m)$

The degree of the simple extension K a over K .

definition *ext-degree* :: $'a :: \text{field set} \Rightarrow 'a \Rightarrow \text{nat}$
where *ext-degree* K $a = \text{degree } (\text{minpoly } K \text{ } a)$

context *Subfield*
begin

For an algebraic element, *minpoly* genuinely is a minimal polynomial.

lemma *is-minpoly-minpoly*:
assumes *algebraic-over* K a
shows *is-minpoly* K a (*minpoly* K a)
 $\langle proof \rangle$

... and it is *the* minimal polynomial: any minimal polynomial equals it.

lemma *minpoly-eq*:
assumes *algebraic-over* K a **and** *is-minpoly* K a m
shows *minpoly* K $a = m$
 $\langle proof \rangle$

lemma *minpoly-over*: *algebraic-over* K $a \implies \text{minpoly } K \text{ } a \in \text{poly-over } K$
 $\langle proof \rangle$

lemma *minpoly-monic*: *algebraic-over* K $a \implies \text{lead-coeff } (\text{minpoly } K \text{ } a) = 1$
 $\langle proof \rangle$

lemma *minpoly-root*: *algebraic-over* K $a \implies \text{poly } (\text{minpoly } K \text{ } a) \text{ } a = 0$
 $\langle proof \rangle$

lemma *minpoly-nonzero*:
assumes *algebraic-over* K a **shows** *minpoly* K $a \neq 0$
 $\langle proof \rangle$

76.3 The extension degree

The degree of a simple algebraic extension is positive.

lemma *ext-degree-pos*:
assumes *algebraic-over K a*
shows *ext-degree K a > 0*
 $\langle \text{proof} \rangle$

For an element already in K , its minimal polynomial is the linear $[-a, 1:]$.

lemma *is-minpoly-linear*:
assumes *a: a ∈ K*
shows *is-minpoly K a [-a, 1:]*
 $\langle \text{proof} \rangle$

Consequently an element of K is algebraic and has extension degree 1.

lemma *algebraic-over-self*:
assumes *a ∈ K*
shows *algebraic-over K a*
 $\langle \text{proof} \rangle$

lemma *minpoly-in-field*:
assumes *a ∈ K*
shows *minpoly K a = [-a, 1:]*
 $\langle \text{proof} \rangle$

lemma *ext-degree-in-field*:
assumes *a ∈ K*
shows *ext-degree K a = 1*
 $\langle \text{proof} \rangle$

Degree one characterises membership. An algebraic element has extension degree 1 exactly when it already lies in K : a monic degree-one minimal polynomial is $[c, 1:]$ with $c \in K$, and vanishing at a forces $a = -c \in K$.

lemma *ext-degree-eq-1-iff*:
assumes *algebraic-over K a*
shows *ext-degree K a = 1 $\longleftrightarrow$ a ∈ K*
 $\langle \text{proof} \rangle$

poly-over is closed under finite sums (needed to build a polynomial from a coordinate function).

lemma *poly-over-sum*:
assumes $\bigwedge i. i \in A \implies f\ i \in \text{poly-over } K$
shows $(\sum i \in A. f\ i) \in \text{poly-over } K$
 $\langle \text{proof} \rangle$

76.4 The power basis of a simple algebraic extension

Spanning. Every element of the simple extension $\text{eval-img } K\ a$ is a K -linear combination of the powers $1, a, \dots, a^{\text{ext-degree } K\ a - 1}$. Given $y = \text{poly } p\ a$,

divide p by the (monic) minimal polynomial $\text{minpoly } K a$; the quotient term vanishes at a (as $\text{poly } (\text{minpoly } K a) a = 0$), leaving a remainder of degree $< d$ whose coefficients supply the combination.

lemma *power-basis-span:*

assumes *alg:* algebraic-over $K a$ **and** $y: y \in \text{eval-img } K a$

shows $\exists c. (\forall i < \text{ext-degree } K a. c i \in K) \wedge y = (\sum i < \text{ext-degree } K a. c i * a^i)$
<proof>

Independence. The powers $1, a, \dots, a^{\text{ext-degree } K a - 1}$ are K -linearly independent: a vanishing combination has all coefficients 0 . The combination is $\text{poly } r a$ for $r = (\sum i < d. \text{monom } (c i) i)$, a polynomial over K of degree $< d$; were it nonzero it would annihilate a below the minimal degree, so $r = 0$ and every coefficient $c i$ vanishes.

lemma *power-basis-indep:*

assumes *alg:* algebraic-over $K a$

and $cK: \bigwedge i. i < \text{ext-degree } K a \implies c i \in K$

and *zero:* $(\sum i < \text{ext-degree } K a. c i * a^i) = 0$

shows $\forall i < \text{ext-degree } K a. c i = 0$

<proof>

end

context *subfield-tower*

begin

For a simple algebraic extension, the vector-space dimension is the existing minimal-polynomial degree. The proof turns the power-basis spanning and independence results into the coordinate-map basis used by *vs.dimension*; the explicit equality is the API bridge between the two presentations of extension degree.

theorem *simple-extension-dimension-eq-ext-degree:*

assumes *alg:* algebraic-over $K a$ **and** *L-def:* $L = \text{eval-img } K a$

shows $\text{vs.dimension} = \text{ext-degree } K a$

<proof>

end

76.5 The tower law, for a two-step simple tower

For a two-step tower $K \subseteq K(a) \subseteq K(a)(b)$ the products $a^i b^j$ (with i below $\text{ext-degree } K a$ and j below $\text{ext-degree } (\text{eval-img } K a) b$) form a K -basis of $K(a)(b)$: the degrees *multiply*. We prove the two basis properties directly from the power bases of the two simple steps, so no dimension-uniqueness (Steinitz) lemma is needed.

Spanning. Every element of $K(a)(b)$ is a K -combination of the products $a^i b^j$. Expand along the power basis of $K(a)(b)$ over $K(a)$, then expand each $K(a)$ -coordinate along the power basis of $K(a)$ over K .

lemma *tower-power-basis-span:*

assumes K : Subfield K
and $\text{alg-}a$: algebraic-over K a
and $\text{alg-}b$: algebraic-over $(\text{eval-}\text{img } K \ a) \ b$
and y : $y \in \text{eval-}\text{img } (\text{eval-}\text{img } K \ a) \ b$
shows $\exists c. (\forall i < \text{ext-degree } K \ a. \forall j < \text{ext-degree } (\text{eval-}\text{img } K \ a) \ b. c \ i \ j \in K) \wedge$
 $y = (\sum j < \text{ext-degree } (\text{eval-}\text{img } K \ a) \ b. \sum i < \text{ext-degree } K \ a. c \ i \ j * (a \wedge^i * b \wedge^j))$
 $\langle \text{proof} \rangle$

Independence. A vanishing K -combination of the products $a^i b^j$ is trivial. Factor b^j out of the inner sum: the coefficients of the b -powers are E -elements, so they vanish by independence over E ; each is itself a K -combination of the a -powers, so its coefficients vanish by independence over K .

lemma *tower-power-basis-indep:*

assumes K : Subfield K
and $\text{alg-}a$: algebraic-over K a
and $\text{alg-}b$: algebraic-over $(\text{eval-}\text{img } K \ a) \ b$
and cK : $\bigwedge i \ j. i < \text{ext-degree } K \ a \implies j < \text{ext-degree } (\text{eval-}\text{img } K \ a) \ b \implies c \ i \ j \in K$
and zero : $(\sum j < \text{ext-degree } (\text{eval-}\text{img } K \ a) \ b. \sum i < \text{ext-degree } K \ a. c \ i \ j * (a \wedge^i * b \wedge^j)) = 0$
shows $\forall i < \text{ext-degree } K \ a. \forall j < \text{ext-degree } (\text{eval-}\text{img } K \ a) \ b. c \ i \ j = 0$
 $\langle \text{proof} \rangle$

end

77 Finite field extensions and their degree

theory *Finite-Extension*

imports *Extension-Degree*

begin

A finite extension is a finitely spanned field tower in the native set-based representation. The underlying vector space already supplies the basis and dimension machinery; this theory gives that machinery a field-theoretic name and keeps the extension degree independent of a choice of basis.

locale *finite-subfield-tower* = *subfield-tower* +

assumes *finite-basis*: $\exists B. \text{vs.basis } B$

begin

definition *extension-degree* :: *nat*

where *extension-degree* $\equiv \text{vs.dimension}$

lemma *extension-degree-eq-dimension:*
extension-degree = vs.dimension
 ⟨proof⟩

lemma *extension-degree-eq-card-basis:*
assumes *vs.basis B*
shows *extension-degree = card B*
 ⟨proof⟩

lemma *finite-basis-exists: $\exists B. \text{vs.basis } B$*
 ⟨proof⟩

The degree is positive because the ambient field contains distinct zero and one. This proof does not require the carrier of the extension to be finite: finite-dimensionality is exactly the basis premise of the locale.

lemma *extension-degree-pos: extension-degree > 0*
 ⟨proof⟩

A simple algebraic extension is finite, and its vector-space dimension is the degree of the minimal polynomial. The substantive basis argument lives in the extension-degree theory; this corollary is the stable finite-extension-facing entry point.

lemma *extension-degree-eq-ext-degree:*
assumes *alg: algebraic-over K a* **and** *L-def: $L = \text{eval-img } K a$*
shows *extension-degree = ext-degree K a*
 ⟨proof⟩

lemma *finite-extension-cardinality:*
assumes *finite L*
shows *card L = card K $\wedge$ extension-degree*
 ⟨proof⟩

A finite-dimensional extension of a finite carrier field again has finite carrier. A basis identifies the extension bijectively with the finitely supported coordinate functions on that basis; when both the basis and coefficient carrier are finite, so is that function space.

lemma *finite-carrier-of-finite-base:*
assumes *finK: finite K*
shows *finite L*
 ⟨proof⟩

Every vector-space basis is also a field-generating set. One inclusion is minimality of *generate-field*; the other expands an arbitrary vector in its basis coordinates.

lemma *generate-field-basis:*
assumes *basis: vs.basis B*
shows *$L = \text{generate-field } (K \cup B)$*
 ⟨proof⟩

end

context *subfield-tower*
begin

Finite spanning is the representation-independent introduction rule for the finite-extension locale. It is deliberately phrased using the existing span API, so callers need not manufacture a basis merely to establish finite-dimensionality.

lemma *finite-subfield-tower-of-finite-spanning*:
 assumes *finB*: *finite B* **and** *BL*: $B \subseteq L$ **and** *span*: $L \subseteq \text{vs.span } B$
 shows *finite-subfield-tower K L*
 $\langle \text{proof} \rangle$

lemma *finite-subfield-tower-iff-finite-spanning*:
 $\text{finite-subfield-tower } K L \longleftrightarrow$
 $(\exists B. \text{finite } B \wedge B \subseteq L \wedge L \subseteq \text{vs.span } B)$
 $\langle \text{proof} \rangle$

end

77.1 Finite intermediate extensions

A subfield between the endpoints of a finite extension is finite-dimensional over the base. It is a vector subspace of the ambient finite-dimensional extension, so the subspace basis theorem supplies a finite basis.

theorem *finite-subfield-tower-intermediate-left*:
 fixes $F E K :: 'a :: \text{field set}$
 assumes *T*: *finite-subfield-tower F K*
 and *sfE*: *Subfield E* **and** *FE*: $F \subseteq E$ **and** *EK*: $E \subseteq K$
 shows *finite-subfield-tower F E*
 $\langle \text{proof} \rangle$

The upper part of the same tower is finite-dimensional as well. An ambient basis over the smaller base still spans after scalars are enlarged to the intermediate field.

theorem *finite-subfield-tower-intermediate-right*:
 fixes $F E K :: 'a :: \text{field set}$
 assumes *T*: *finite-subfield-tower F K*
 and *sfE*: *Subfield E* **and** *FE*: $F \subseteq E$ **and** *EK*: $E \subseteq K$
 shows *finite-subfield-tower E K*
 $\langle \text{proof} \rangle$

end

78 Finite-dimensional extensions and transitivity of algebraicity

```
theory Algebraic-Transitivity
  imports Finite-Extension
begin
```

With the vector-space bridge of *Extension-Vector-Space* in place, the Steinitz cardinality bound *Vector-Space.independent-le-span* yields the classical chain

- a finite-dimensional extension is *algebraic* (every element satisfies a polynomial: its powers cannot all be independent);
- *transitivity of algebraicity* — if a is algebraic over K and b is algebraic over the simple extension $K(a)$, then b is algebraic over K .

This is the last field-theoretic prerequisite of the quintic finish: it upgrades “a radical is algebraic over the immediate base” to “a radical is algebraic over the ground field”, so the normal closure can be taken as the splitting field of a single ground-field polynomial.

```
notation plus (infixl <+> 65)
notation minus (infixl <-> 65)
unbundle uminus-syntax
```

78.1 A finite-dimensional extension is algebraic

If the extension L of K is spanned over K by a finite set B , then every element of L is algebraic over K . For $y \in L$ the $\text{card } B + 1$ powers y^i (with $i \leq \text{card } B$) all lie in the span of B , so by the Steinitz bound they are linearly dependent: some nontrivial K -combination vanishes, i.e. a nonzero polynomial over K kills y .

```
lemma (in subfield-tower) findim-imp-algebraic:
  assumes finB: finite B and BL: B ⊆ L and span: L ⊆ vs.span B and y: y ∈ L
  shows algebraic-over K y
  <proof>
```

Finite-dimensionality makes every target element algebraic over the base. This bridge belongs with the finite-dimensional algebraicity theorem, so clients need not import primitive-element constructions merely to use it.

```
lemma (in finite-subfield-tower) finite-extension-algebraic:
  assumes aL: a ∈ L
  shows algebraic-over K a
  <proof>
```

78.2 Transitivity of algebraicity

Transitivity. If a is algebraic over K and b is algebraic over the simple extension $\text{eval-img } K \ a$, then b is algebraic over K .

The two-step tower $K \subseteq K(a) \subseteq K(a)(b)$ is spanned over K by the finite set of products $a^i * b^j$ (*tower-power-basis-span*); so $K(a)(b)$ is a finite-dimensional extension of K , and b , being one of its elements, is algebraic over K by *subfield-tower.findim-imp-algebraic*.

theorem *algebraic-over-transitive:*

assumes K : Subfield K
and $alg-a$: algebraic-over K a
and $alg-b$: algebraic-over ($eval-imag$ K a) b
shows algebraic-over K b

<proof>

78.3 Algebraicity along a radical/simple tower

Every element of a simple algebraic extension $eval-imag$ K a is itself algebraic over K : the extension is spanned over K by the finite power basis a^i (*Subfield.power-basis-span*), so *subfield-tower.findim-imp-algebraic* applies.

theorem *simple-ext-algebraic:*

assumes K : Subfield K **and** $alg-a$: algebraic-over K a **and** y : $y \in eval-imag$ K a
shows algebraic-over K y

<proof>

Two successive simple algebraic extensions are algebraic over the original base. This interface is useful when proving closure of the algebraic elements under field operations: the operation is first placed in the two-step generated field, and this lemma supplies its algebraicity over the base.

lemma *algebraic-over-two-step:*

fixes F :: 'a :: field set **and** a b y :: 'a
assumes sfF : Subfield F
and $alg-a$: algebraic-over F a
and $alg-b$: algebraic-over F b
and y : $y \in eval-imag$ ($eval-imag$ F a) b
shows algebraic-over F y

<proof>

definition *algebraic-elements* :: 'a :: field set $\Rightarrow$ 'a set **where**

algebraic-elements $F = \{a. \text{algebraic-over } F \ a\}$

Algebraic elements over a subfield form a subfield. Addition and multiplication are handled through *algebraic-over-two-step*; negation and inversion already lie in the corresponding simple extension.

lemma *subfield-algebraic-elements:*

fixes F :: 'a :: field set
assumes sfF : Subfield F
shows Subfield (*algebraic-elements* F)

<proof>

Generation by algebraic elements preserves algebraicity. The generated field is

contained in the subfield of all elements algebraic over the base, so no representation of generated elements or basis choice is exposed to clients.

```

lemma generated-algebraic:
  fixes  $F A :: 'a :: \text{field set}$ 
  assumes  $\text{sf}F$ :  $\text{Subfield } F$ 
  and  $\text{alg}A$ :  $\bigwedge a. a \in A \implies \text{algebraic-over } F a$ 
  and  $x$ :  $x \in \text{generate-field } (F \cup A)$ 
  shows  $\text{algebraic-over } F x$ 
   $\langle \text{proof} \rangle$ 

end

```

79 Generic properties of algebraic field extensions

```

theory Extension-Properties
  imports Extension-Degree
begin

```

These are the representation-independent extension predicates used by the Galois development. A field extension is represented by nested set-based subfields of one ambient field, as in *Subfield*; no complex-specific subfield locale or algebraic-closure theorem is needed for the definitions.

A splitting field is presented by a nonzero polynomial over the base and the subfield generated by all of its roots. The normality predicate uses the same root-set presentation for every irreducible polynomial over the base, while separability is stated through minimal polynomials.

```

definition poly-root-set ::  $'a :: \text{field poly} \Rightarrow 'a \text{ set}$  where
  poly-root-set  $p \equiv \{z. \text{poly } p z = 0\}$ 

```

```

definition splitting-field ::
   $'a :: \text{field set} \Rightarrow 'a \text{ poly} \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$  where
  splitting-field  $F p K \equiv$ 
     $p \in \text{poly-over } F \wedge p \neq 0 \wedge K = \text{generate-field } (F \cup \text{poly-root-set } p)$ 

```

```

definition algebraic-extension ::  $'a :: \text{field set} \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$  where
  algebraic-extension  $K F \equiv (\forall a \in K. \text{algebraic-over } F a)$ 

```

```

definition normal-extension ::  $'a :: \text{field set} \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$  where
  normal-extension  $K F \equiv$ 
     $(\forall p a. p \in \text{poly-over } F \longrightarrow \text{irreducible-over } F p \longrightarrow a \in K \longrightarrow$ 
       $\text{poly } p a = 0 \longrightarrow \text{poly-root-set } p \subseteq K)$ 

```

```

definition separable-extension ::  $'a :: \text{field set} \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$  where
  separable-extension  $K F \equiv$ 
     $(\forall a \in K. \text{algebraic-over } F a \longrightarrow \text{rsquarefree } (\text{minpoly } F a))$ 

```

lemma *poly-root-set-iff* [simp]:

$z \in \text{poly-root-set } p \iff \text{poly } p \ z = 0$
 $\langle \text{proof} \rangle$

lemma *finite-poly-root-set*:

$p \neq 0 \implies \text{finite } (\text{poly-root-set } p)$
 $\langle \text{proof} \rangle$

lemma *splitting-fieldD*:

$\text{splitting-field } F \ p \ K \implies$
 $p \in \text{poly-over } F \wedge p \neq 0 \wedge K = \text{generate-field } (F \cup \text{poly-root-set } p)$
 $\langle \text{proof} \rangle$

lemma

assumes *splitting-field* $F \ p \ K$
shows *splitting-field-subfield*: $\text{Subfield } K$
and *splitting-field-base-subset*: $F \subseteq K$
and *splitting-field-roots-subset*: $\text{poly-root-set } p \subseteq K$
 $\langle \text{proof} \rangle$

lemma *algebraic-extensionI*:

assumes $\bigwedge a. a \in K \implies \text{algebraic-over } F \ a$
shows *algebraic-extension* $K \ F$
 $\langle \text{proof} \rangle$

lemma *algebraic-extensionD*:

assumes *algebraic-extension* $K \ F$ **and** $a \in K$
shows *algebraic-over* $F \ a$
 $\langle \text{proof} \rangle$

lemma *normal-extensionI*:

assumes
 $\bigwedge p \ a. [p \in \text{poly-over } F; \text{irreducible-over } F \ p; a \in K; \text{poly } p \ a = 0] \implies \text{poly-root-set } p \subseteq K$
shows *normal-extension* $K \ F$
 $\langle \text{proof} \rangle$

lemma *normal-extensionD*:

assumes *normal-extension* $K \ F$
and $p \in \text{poly-over } F \text{ irreducible-over } F \ p \ a \in K \text{ poly } p \ a = 0$
shows $\text{poly-root-set } p \subseteq K$
 $\langle \text{proof} \rangle$

lemma *separable-extensionI*:

assumes $\bigwedge a. a \in K \implies \text{algebraic-over } F \ a \implies \text{rsquarefree } (\text{minpoly } F \ a)$
shows *separable-extension* $K \ F$
 $\langle \text{proof} \rangle$

lemma *separable-extensionD*:

assumes *separable-extension* $K \ F$ **and** $a \in K$ **and** *algebraic-over* $F \ a$

```

shows rsquarefree (minpoly F a)
  ⟨proof⟩

```

A divisor of a root-squarefree polynomial is root-squarefree. This is the base-change bridge used below: a minimal polynomial over a larger subfield divides the minimal polynomial over the smaller one, so separability descends to the larger base.

```

lemma rsquarefree-dvd:
  fixes p q :: 'a :: idom poly
  assumes dvd: p dvd q and sq: rsquarefree q
  shows rsquarefree p
  ⟨proof⟩

```

Over an algebraically closed field, root-squarefreeness turns the multiset of polynomial roots into an ordinary set of the same cardinality.

```

lemma card-roots-eq-degree-alg-closed:
  fixes p :: 'a :: alg-closed-field poly
  assumes p0: p ≠ 0 and rsf: rsquarefree p
  shows card {x. poly p x = 0} = degree p
  ⟨proof⟩

```

A root of the defining polynomial of a splitting field is algebraic over its base.

```

lemma splitting-field-root-algebraic:
  assumes split: splitting-field F p K
  and r: r ∈ poly-root-set p
  shows algebraic-over F r
  ⟨proof⟩

```

end

80 The tower law for finite field extensions

```

theory Finite-Extension-Tower
  imports Finite-Extension
begin

```

A pair of finite field extensions $K \subseteq L \subseteq M$ carries three native vector-space interpretations. This theory proves that a basis of L over K and a basis of M over L multiply to a basis of M over K . The resulting degree formula is independent of the chosen bases and is the reusable tower-law interface for later Galois work.

```

locale finite-subfield-tower-chain =
  KL: finite-subfield-tower K L + LM: finite-subfield-tower L M
  for K L M :: 'a :: field set
begin

```

```

lemma subfield-tower-KM: subfield-tower K M

```

$\langle \text{proof} \rangle$

sublocale *KM*: *subfield-tower* *K M*

$\langle \text{proof} \rangle$

definition *product-map* :: $'a \times 'a \Rightarrow 'a$

where *product-map* $z = \text{snd } z * \text{fst } z$

definition *product-basis* :: $'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set}$

where *product-basis* $B \ C = \{b * c \mid b \in B \wedge c \in C\}$

lemma *product-basis-image*:

product-basis $B \ C = \text{product-map } ' (C \times B)$

$\langle \text{proof} \rangle$

lemma *KL-basis-subset*:

assumes *KL.vs.basis* B

shows $B \subseteq L$

$\langle \text{proof} \rangle$

lemma *LM-basis-subset*:

assumes *LM.vs.basis* C

shows $C \subseteq M$

$\langle \text{proof} \rangle$

lemma *KM-product-map-inj*:

assumes B : *KL.vs.basis* B **and** C : *LM.vs.basis* C

shows *inj-on* *product-map* $(C \times B)$

$\langle \text{proof} \rangle$

lemma *KM-product-basis-card*:

assumes B : *KL.vs.basis* B **and** C : *LM.vs.basis* C

shows $\text{card } (\text{product-basis } B \ C) = \text{card } B * \text{card } C$

$\langle \text{proof} \rangle$

lemma *KM-product-basis-subset*:

assumes B : *KL.vs.basis* B **and** C : *LM.vs.basis* C

shows *product-basis* $B \ C \subseteq M$

$\langle \text{proof} \rangle$

lemma *KM-product-basis-spanning*:

assumes B : *KL.vs.basis* B **and** C : *LM.vs.basis* C

shows $M \subseteq \text{KM.vs.span } (\text{product-basis } B \ C)$

$\langle \text{proof} \rangle$

lemma *KM-product-basis-condition*:

assumes B : *KL.vs.basis* B **and** C : *LM.vs.basis* C

shows $\forall f \in \text{product-basis } B \ C \rightarrow_E K.$

$\text{KM.vs.lincomb } f \ (\text{product-basis } B \ C) = 0 \longrightarrow$

$(\forall v \in \text{product-basis } B \ C. f \ v = 0)$

<proof>

lemma *KM-product-basis-independent:*

assumes *B: KL.vs.basis B and C: LM.vs.basis C*

shows *KM.vs.lin-indep (product-basis B C)*

<proof>

theorem *KM-product-basis:*

assumes *B: KL.vs.basis B and C: LM.vs.basis C*

shows *KM.vs.basis (product-basis B C)*

<proof>

sublocale *KMF: finite-subfield-tower K M*

<proof>

theorem *extension-degree-tower-law:*

assumes *B: KL.vs.basis B and C: LM.vs.basis C*

shows *KMF.extension-degree = KL.extension-degree * LM.extension-degree*

<proof>

theorem *extension-degree-tower-law-exists:*

*KMF.extension-degree = KL.extension-degree * LM.extension-degree*

<proof>

end

end

81 Extensions generated by finitely many algebraic elements

theory *Finite-Generated-Extension*

imports *Finite-Extension-Tower*

begin

Adjoining one algebraic element to a subfield gives a finite extension — the power basis $1, a, \dots, a^{d-1}$ spans it — and finite extensions compose. Adjoining finitely many algebraic elements one at a time therefore stays finite. This is HOL-Algebra's *finite-extension*, and it is what turns a polynomial with algebraic coefficients into a polynomial over a finite extension of the base field, so that a dimension count can be applied to it.

The pieces were all present: *power-basis-span* supplies the spanning set, *sub-field-tower.finite-subfield-tower-of-finite-spanning* converts it into finite-dimensionality, and the chain locale composes two finite extensions. Only the assembly was missing.

81.1 A subfield is a finite extension of itself

lemma *finite-subfield-tower-self*:
assumes *sfK*: Subfield *K*
shows *finite-subfield-tower* *K K*
 ⟨*proof*⟩

81.2 A simple algebraic extension is finite

The spanning set is the power basis, and the coefficients supplied by *power-basis-span* are fed to *subfield-tower.sum-scale-in-span* unchanged.

lemma *finite-subfield-tower-simple*:
assumes *sfK*: Subfield *K* **and** *alg*: algebraic-over *K a*
shows *finite-subfield-tower* *K (eval-img K a)*
 ⟨*proof*⟩

81.3 Finite extensions compose

lemma *finite-subfield-tower-trans*:
assumes *KL*: *finite-subfield-tower* *K L* **and** *LM*: *finite-subfield-tower* *L M*
shows *finite-subfield-tower* *K M*
 ⟨*proof*⟩

81.4 Finitely many algebraic elements generate a finite extension

Adjoin the elements one at a time. Each is algebraic over the base and hence, by *algebraic-over-mono*, over every intermediate field, so the simple-extension result applies at each step and the composition keeps the whole tower finite.

theorem *exists-finite-subfield-tower*:
assumes *sfK*: Subfield *K* **and** *finS*: finite *S*
and *algS*: $\bigwedge a. a \in S \implies \text{algebraic-over } K a$
shows $\exists L. \text{finite-subfield-tower } K L \wedge K \subseteq L \wedge S \subseteq L$
 ⟨*proof*⟩

81.5 Coordinates over a finite extension

A finite extension carries a finite spanning set together with a coordinate function writing each of its elements as an explicit combination over the base. Stated inside the locale, where the two fields are parameters: a client whose base field is a described set — an image, say — would otherwise send the automatic methods hunting through that description.

lemma (in *finite-subfield-tower*) *exists-coordinates*:
obtains *B c* **where** *finite B* **and** $B \subseteq L$
and $\bigwedge x. x \in L \implies (\forall b \in B. c\ x\ b \in K)$
and $\bigwedge x. x \in L \implies x = (\sum b \in B. c\ x\ b * b)$
 ⟨*proof*⟩

end

82 Homogeneous linear systems with more unknowns than equations

theory *Linear-System*
imports *Subfield*
begin

A homogeneous linear system with strictly more unknowns than equations has a nontrivial solution. This is the pigeonhole principle of linear algebra, and it is what powers *Artin's theorem* on the fixed field of a finite group of automorphisms: there one produces a nonzero solution of the system $\sum_j \sigma(x_j) c_j = 0$, one equation per automorphism σ , from an assumed excess of independent elements x_j .

We index equations by a finite set I and unknowns by a finite set J , rather than by initial segments $\{..<m\}$ and $\{..<n\}$. This matters: the proof eliminates one unknown at each step, and with index *sets* that is just $J - \{k\}$, whereas with initial segments every step would need a reindexing bijection.

Everything happens inside a subfield K : the coefficients lie in K and so does the solution, since Gaussian elimination only ever adds, multiplies and divides.

context *Subfield*
begin

The proof is Gaussian elimination, by induction on the set of equations.

With no equations at all, any nonzero vector will do — and there is an unknown to put it in, since $\text{card } \{\} < \text{card } J$ forces J nonempty.

For the inductive step, single out one equation. If all its coefficients vanish it is no constraint, and the induction hypothesis applied to the remaining equations already gives a solution. Otherwise some coefficient $A\ i_0\ k$ is nonzero, and we use that equation to *eliminate* the unknown k : subtracting a suitable multiple of equation i_0 from each remaining equation produces a smaller system in the unknowns $J - \{k\}$, to which the induction hypothesis applies because $\text{card } I < \text{card } J - 1$. Solving that and then choosing $c\ k$ to satisfy equation i_0 solves the whole system. Nontriviality survives because the recursively obtained solution is already nonzero *somewhere in* $J - \{k\}$, and we do not touch those entries.

theorem *underdetermined-solution*:

fixes $A :: 'i \Rightarrow 'j \Rightarrow 'a$
assumes *finite I and finite J and card I < card J*
and $\bigwedge i\ j. \llbracket i \in I; j \in J \rrbracket \implies A\ i\ j \in K$
shows $\exists c. (\forall j. c\ j \in K) \wedge (\exists j \in J. c\ j \neq 0) \wedge (\forall i \in I. (\sum j \in J. A\ i\ j * c\ j) = 0)$
 $\langle \text{proof} \rangle$

end

end

83 The Algebraic Closure of a Field

theory *Closure-Algebraic*

imports *Closure-Extension Poly-Typeclass Algebraic-Transitivity Extension-Properties*

Finite-Generated-Extension Linear-System

HOL-Computational-Algebra.Polynomial-Factorial HOL-Computational-Algebra.Field-as-Ring

begin

Closure-Extension produces, for any field, an extension inside the indexed polynomial ring in which every polynomial over the base field splits. That extension is not yet an algebraic closure: it is far too big, being generated by one indeterminate for every polynomial and root index, and it is not algebraic over the base. The closure is cut out of it as the set of elements algebraic over the base. Rather than redo the theory of algebraicity over a subfield for locale fields, we transport the extension onto a type. The development's field-extension and Galois theory already works with an ambient type-class field and subfields as carrier sets, and once the extension is a type all of it applies — in particular *subfield-algebraic-elements*, which is the statement that the algebraic elements form a subfield.

83.1 An extension of the whole type in which every polynomial splits

context *poly-TC*

begin

A definite choice among the extensions supplied by *exists-extension-with-roots*, so that the type constructed from it below depends on nothing but *'a*.

definition *wide-extension*

where *wide-extension* $\equiv$

(*SOME* *M*. *is-extension* *M*

$\wedge (\forall P \in \text{poly-carrier}.$

Field.splits (*ecarrier* *M*) (*eadd* *M*) (*emult* *M*) *ipoly-zero ipoly-one*

(*lift-poly* *P*)))

lemma *wide-extension-spec*:

is-extension *wide-extension*

$\wedge (\forall P \in \text{poly-carrier}.$

Field.splits (*ecarrier* *wide-extension*) (*eadd* *wide-extension*) (*emult* *wide-extension*)

ipoly-zero ipoly-one (*lift-poly* *P*))

$\langle \text{proof} \rangle$

lemmas *is-extension-wide-extension* = *wide-extension-spec* [*THEN* *conjunct1*]

lemma *wide-extension-splits*:

$P \in \text{poly-carrier}$
 $\implies \text{Field.splits } (\text{ecarrier wide-extension}) (\text{eadd wide-extension}) (\text{emult wide-extension})$
 $\text{ipoly-zero ipoly-one (lift-poly } P)$
 $\langle \text{proof} \rangle$

The field structure of the chosen extension, named once for the transport below.

lemma *field-wide-extension*:

$\text{Field } (\text{ecarrier wide-extension}) (\text{eadd wide-extension}) (\text{emult wide-extension})$
 $\text{ipoly-zero ipoly-one}$
 $\langle \text{proof} \rangle$

lemma *wide-extension-nonempty*: $\text{ipoly-zero} \in \text{ecarrier wide-extension}$
 $\langle \text{proof} \rangle$

How the base field sits inside the extension. These are restatements of the extension conditions at the chosen extension: only facts proved *in this locale* are available once it is left, and the type constructed below can only be spoken of outside it.

lemma *wide-const-mem*: $\text{ipoly-const } a \in \text{ecarrier wide-extension}$
 $\langle \text{proof} \rangle$

lemma *wide-const-add*:

$\text{eadd wide-extension } (\text{ipoly-const } a) (\text{ipoly-const } b) = \text{ipoly-const } (a + b)$
 $\langle \text{proof} \rangle$

lemma *wide-const-mult*:

$\text{emult wide-extension } (\text{ipoly-const } a) (\text{ipoly-const } b) = \text{ipoly-const } (a * b)$
 $\langle \text{proof} \rangle$

lemmas *wide-const-inj* = *ipoly-const-inj-on*

lemmas *wide-lift-poly-apply* = *lift-poly-apply*

lemmas *wide-const-zero* = *ipoly-const-zero*

lemmas *wide-const-one* = *ipoly-const-one*

end

83.2 The extension as a type

The extension is a field, but a locale one, on a carrier set inside the indexed polynomial ring. Carving that carrier out as a type and transporting the operations onto it makes it a type-class field, which is what the rest of the development can use.

typedef (**overloaded**) $'a :: \text{field wide} = \text{ecarrier } (\text{poly-TC.wide-extension} :: 'a \text{ iext})$
 $\langle \text{proof} \rangle$

setup-lifting *type-definition-wide*

Names for the transported structure. The locale's constants must be spelled out here because *poly-TC* fixes no parameters, so its abbreviations do not reach outside it; each takes only the parameters occurring in its definition.

abbreviation *wcarrier* :: 'a :: field ipoly set
where *wcarrier* $\equiv$ *ecarrier* (*poly-TC.wide-extension* :: 'a iext)

abbreviation *wadd* :: 'a :: field ipoly $\Rightarrow$ 'a ipoly $\Rightarrow$ 'a ipoly
where *wadd* $\equiv$ *eadd* (*poly-TC.wide-extension* :: 'a iext)

abbreviation *wmult* :: 'a :: field ipoly $\Rightarrow$ 'a ipoly $\Rightarrow$ 'a ipoly
where *wmult* $\equiv$ *emult* (*poly-TC.wide-extension* :: 'a iext)

abbreviation *wzero* :: 'a :: field ipoly
where *wzero* $\equiv$ *Ring.ipoly-zero* 0

abbreviation *wone* :: 'a :: field ipoly
where *wone* $\equiv$ *Ring.ipoly-one* 0 1

abbreviation *wneg* :: 'a :: field ipoly $\Rightarrow$ 'a ipoly
where *wneg* $\equiv$ *Monoid.inverse* *wcarrier* *wadd* *wzero*

abbreviation *winv* :: 'a :: field ipoly $\Rightarrow$ 'a ipoly
where *winv* $\equiv$ *Monoid.inverse* *wcarrier* *wmult* *wone*

interpretation *W*: *Field* *wcarrier* *wadd* *wmult* *wzero* *wone*
 $\langle$ *proof* $\rangle$

instantiation *wide* :: (field) field
begin

lift-definition *zero-wide* :: 'a wide **is** *wzero* :: 'a ipoly
 $\langle$ *proof* $\rangle$

lift-definition *one-wide* :: 'a wide **is** *wone* :: 'a ipoly
 $\langle$ *proof* $\rangle$

lift-definition *plus-wide* :: 'a wide $\Rightarrow$ 'a wide $\Rightarrow$ 'a wide **is** *wadd* :: 'a ipoly $\Rightarrow$ -
 $\langle$ *proof* $\rangle$

lift-definition *times-wide* :: 'a wide $\Rightarrow$ 'a wide $\Rightarrow$ 'a wide **is** *wmult* :: 'a ipoly $\Rightarrow$ -
 $\langle$ *proof* $\rangle$

lift-definition *uminus-wide* :: 'a wide $\Rightarrow$ 'a wide **is** *wneg* :: 'a ipoly $\Rightarrow$ -
 $\langle$ *proof* $\rangle$

lift-definition *minus-wide* :: 'a wide $\Rightarrow$ 'a wide $\Rightarrow$ 'a wide
is $\lambda x y. wadd\ x\ (wneg\ y)$:: 'a ipoly
 $\langle$ *proof* $\rangle$

HOL's field class requires *inverse* $0 = 0$, where the locale leaves the inverse of *wzero* unspecified; the two are reconciled by the case split.

lift-definition *inverse-wide* :: 'a wide $\Rightarrow$ 'a wide
is $\lambda x.$ if $x = wzero$ then $wzero$ else $winv\ x$:: 'a ipoly
 <proof>

lift-definition *divide-wide* :: 'a wide $\Rightarrow$ 'a wide $\Rightarrow$ 'a wide
is $\lambda x\ y.$ if $y = wzero$ then $wzero$ else $wmult\ x\ (winv\ y)$:: 'a ipoly
 <proof>

instance
 <proof>

end

A field is a Euclidean ring in a trivial way. Spelling that out for 'a wide — as HOL-Algebra's *Algebraic-Closure-Type* does for its own closure type — is what puts 'a wide poly in *factorial-semiring*, and so makes HOL's factorisation machinery, in particular the primality of a monic linear polynomial, available over the extension.

instantiation *wide* :: (field)
 {unique-euclidean-ring, normalization-euclidean-semiring, normalization-semidom-multiplicative}
begin

definition [simp]: *normalize-wide* = (normalize-field :: 'a wide $\Rightarrow$ -)
definition [simp]: *unit-factor-wide* = (unit-factor-field :: 'a wide $\Rightarrow$ -)
definition [simp]: *modulo-wide* = (mod-field :: 'a wide $\Rightarrow$ -)
definition [simp]: *euclidean-size-wide* = (euclidean-size-field :: 'a wide $\Rightarrow$ -)
definition [simp]: *division-segment* (x :: 'a wide) = 1

instance
 <proof>

end

instantiation *wide* :: (field) euclidean-ring-gcd
begin

definition *gcd-wide* :: 'a wide $\Rightarrow$ 'a wide $\Rightarrow$ 'a wide
where *gcd-wide* = Euclidean-Algorithm.gcd
definition *lcm-wide* :: 'a wide $\Rightarrow$ 'a wide $\Rightarrow$ 'a wide
where *lcm-wide* = Euclidean-Algorithm.lcm
definition *Gcd-wide* :: 'a wide set $\Rightarrow$ 'a wide
where *Gcd-wide* = Euclidean-Algorithm.Gcd
definition *Lcm-wide* :: 'a wide set $\Rightarrow$ 'a wide
where *Lcm-wide* = Euclidean-Algorithm.Lcm

instance
 <proof>

end

instance *wide* :: (*field*) *semiring-gcd-mult-normalize* ⟨*proof*⟩

The four instances above are exactly *field-gcd*, which is the sort under which HOL states the primality of a monic linear polynomial.

instance *wide* :: (*field*) *field-gcd* ⟨*proof*⟩

83.3 The base field inside the extension

The base field sits in the extension as the constants. This is the embedding along which the algebraic closure will be described: a subfield of *'a wide* isomorphic to *'a*.

lift-definition *to-wide* :: *'a* :: *field* $\Rightarrow$ *'a wide* **is** *Ring.ipoly-const 0* :: *'a* $\Rightarrow$ *'a ipoly* ⟨*proof*⟩

lemma *to-wide-zero* [*simp*]: *to-wide 0* = 0
⟨*proof*⟩

lemma *to-wide-one* [*simp*]: *to-wide 1* = 1
⟨*proof*⟩

lemma *to-wide-add* [*simp*]: *to-wide* (*x* + *y*) = *to-wide x* + *to-wide y*
⟨*proof*⟩

lemma *to-wide-mult* [*simp*]: *to-wide* (*x* * *y*) = *to-wide x* * *to-wide y*
⟨*proof*⟩

lemma *inj-to-wide*: *inj* (*to-wide* :: *'a* :: *field* $\Rightarrow$ *'a wide*)
⟨*proof*⟩

lemma *to-wide-eq-iff* [*simp*]: *to-wide x* = *to-wide y* $\longleftrightarrow$ *x* = *y*
⟨*proof*⟩

Being a ring homomorphism between fields, the embedding automatically respects negation and inversion: each is pinned by the equation it solves.

lemma *to-wide-uminus* [*simp*]: *to-wide* (− *x*) = − *to-wide x*
⟨*proof*⟩

lemma *to-wide-inverse* [*simp*]: *to-wide* (*inverse x*) = *inverse* (*to-wide x*)
⟨*proof*⟩

lemma *to-wide-diff* [*simp*]: *to-wide* (*x* − *y*) = *to-wide x* − *to-wide y*
⟨*proof*⟩

theorem *subfield-range-to-wide*: *Subfield* (*range* (*to-wide* :: *'a* :: *field* $\Rightarrow$ *'a wide*))
⟨*proof*⟩

The inverse of the embedding, so that a polynomial over the base subfield can be read back as a polynomial over $'a$.

definition *of-wide* :: $'a :: \text{field wide} \Rightarrow 'a$
where *of-wide* = *inv to-wide*

lemma *of-wide-to-wide* [simp]: *of-wide* (*to-wide* x) = x
 $\langle \text{proof} \rangle$

lemma *to-wide-of-wide* [simp]: $x \in \text{range } \text{to-wide} \implies \text{to-wide } (\text{of-wide } x) = x$
 $\langle \text{proof} \rangle$

lemma *of-wide-zero* [simp]: *of-wide* $0 = 0$
 $\langle \text{proof} \rangle$

83.4 Every polynomial over the base subfield splits

Abs-wide is a field isomorphism from the carrier onto the type, so *splits-hom* carries the splitting of a base polynomial across. Each equation below is read off from the corresponding *rep-eq* of the transported operation.

lemma *Abs-wide-zero* [simp]: *Abs-wide* *wzero* = 0
 $\langle \text{proof} \rangle$

lemma *Abs-wide-one* [simp]: *Abs-wide* *wone* = 1
 $\langle \text{proof} \rangle$

lemma *Abs-wide-add*:
assumes $x \in \text{wcarrier}$ **and** $y \in \text{wcarrier}$
shows *Abs-wide* (*wadd* x y) = *Abs-wide* x + *Abs-wide* y
 $\langle \text{proof} \rangle$

lemma *Abs-wide-mult*:
assumes $x \in \text{wcarrier}$ **and** $y \in \text{wcarrier}$
shows *Abs-wide* (*wmult* x y) = *Abs-wide* x * *Abs-wide* y
 $\langle \text{proof} \rangle$

lemma *Abs-wide-const* [simp]: *Abs-wide* (*Ring.ipoly-const* 0 a) = *to-wide* a
 $\langle \text{proof} \rangle$

A polynomial whose coefficients lie in the base subfield, read back as a polynomial over the base field itself.

definition *wide-pull* :: $'a :: \text{field wide poly} \Rightarrow \text{nat} \Rightarrow 'a$
where *wide-pull* $p \equiv (\lambda k. \text{of-wide } (\text{Polynomial.coeff } p \ k))$

lemma *wide-pull-carrier*: *wide-pull* $p \in \text{Ring.poly-carrier } (\text{UNIV} :: 'a :: \text{field set}) \ 0$
 $\langle \text{proof} \rangle$

lemma *to-wide-wide-pull*:
assumes $p: p \in \text{poly-over } (\text{range } (\text{to-wide} :: 'a :: \text{field} \Rightarrow 'a \text{ wide}))$

shows *to-wide* (*wide-pull* p k) = *Polynomial.coeff* p k
 ⟨*proof*⟩

theorem *wide-splits*:

fixes $p :: 'a :: \text{field wide poly}$

assumes $p :: p \in \text{poly-over } (\text{range to-wide})$

obtains c *as* **where** $p = [:c:] * (\prod a \leftarrow \text{as. } [: \text{uminus } a, 1:])$

⟨*proof*⟩

83.5 The algebraic closure

The splitting extension is far larger than an algebraic closure — it has an indeterminate for every polynomial and root index, and is not algebraic over the base. Cutting it down to the elements algebraic over the base gives the closure. Nothing has to be proved about that set here: *subfield-algebraic-elements* already says it is a subfield, which is exactly what transporting the extension onto a type bought.

definition *alg-closure* :: $'a :: \text{field wide set}$

where *alg-closure* = *algebraic-elements* (*range to-wide*)

theorem *subfield-alg-closure*: *Subfield* (*alg-closure* :: $'a :: \text{field wide set}$)

⟨*proof*⟩

lemma *alg-closure-iff*: $x \in \text{alg-closure} \longleftrightarrow \text{algebraic-over } (\text{range to-wide})\ x$

⟨*proof*⟩

theorem *base-subset-alg-closure*: $\text{range } (\text{to-wide} :: 'a :: \text{field} \Rightarrow 'a \text{ wide}) \subseteq \text{alg-closure}$

⟨*proof*⟩

theorem *algebraic-extension-alg-closure*:

algebraic-extension (*alg-closure* :: $'a :: \text{field wide set}$) (*range to-wide*)

⟨*proof*⟩

Two facts about a product of linear factors: it is monic, and its roots are exactly the elements of the list.

lemma *lead-coeff-linear-prod*: $\text{lead-coeff } (\prod a \leftarrow \text{as. } [: \text{uminus } a, 1:]) = (1 :: 'b :: \text{idom})$

⟨*proof*⟩

lemma *poly-linear-prod-eq-0-iff*:

$\text{poly } (\prod a \leftarrow \text{as. } [: \text{uminus } a, 1:])\ z = (0 :: 'b :: \text{idom}) \longleftrightarrow z \in \text{set as}$

⟨*proof*⟩

A root of a nonzero polynomial over the base is algebraic over it, so lies in the closure — that polynomial is its own witness.

lemma *alg-closure-root*:

fixes $p :: 'a :: \text{field wide poly}$

assumes $p \in \text{poly-over } (\text{range to-wide})$ **and** $p \neq 0$ **and** $\text{poly } p\ z = 0$

shows $z \in \text{alg-closure}$

<proof>

Every nonzero polynomial over the base splits, with all of its roots and its leading coefficient inside the closure. Because *splits* carries an explicit list of roots, the descent from the splitting extension to the closure is immediate: the very same product works, every factor being a root of a base polynomial and hence algebraic.

theorem *alg-closure-splits*:
fixes $p :: 'a :: \text{field wide poly}$
assumes $p \in \text{poly-over (range to-wide)}$ **and** $\text{nz: } p \neq 0$
obtains $c \text{ as}$ **where** $c \in \text{alg-closure}$ **and** $\text{set as} \subseteq \text{alg-closure}$
and $p = [:c:] * (\prod a \leftarrow \text{as. } [:uminus\ a,\ 1:])$
<proof>

corollary *poly-root-set-subset-alg-closure*:
fixes $p :: 'a :: \text{field wide poly}$
assumes $p \in \text{poly-over (range to-wide)}$ **and** $p \neq 0$
shows $\text{poly-root-set } p \subseteq \text{alg-closure}$
<proof>

The existence statement, in the shape of HOL-Algebra's *exists-closure*: every field has an algebraic extension in which each of its polynomials splits.

theorem *exists-closure*:
obtains $L :: 'a :: \text{field wide set}$
where $\text{Subfield } L$ **and** $\text{range to-wide} \subseteq L$ **and** $\text{algebraic-extension } L$ (*range to-wide*)
and $\bigwedge p. \llbracket p \in \text{poly-over (range to-wide)}; p \neq 0 \rrbracket$
 $\implies \exists c \text{ as. } c \in L \wedge \text{set as} \subseteq L \wedge p = [:c:] * (\prod a \leftarrow \text{as. } [:uminus\ a,\ 1:])$
<proof>

83.6 The closure is algebraically closed

A divisor of a split polynomial of positive degree has a root among its factors. Peeling one factor at a time: either it is a root of the divisor, or the divisor is coprime to it (linear monic polynomials are prime) and the factor can be cancelled.

lemma *dvd-split-imp-root*:
fixes $q :: 'b :: \text{field-gcd poly}$
assumes $q \text{ dvd } [: \gamma :] * (\prod a \leftarrow \text{as. } [:uminus\ a,\ 1:])$ **and** $\gamma \neq 0$ **and** $0 < \text{degree } q$
shows $\exists a \in \text{set as. } \text{poly } q\ a = 0$
<proof>

Every polynomial over the closure of positive degree has a root there, so the closure is algebraically closed.

This is the one step of the construction that needs a dimension count, and it is HOL-Algebra's rupture argument in elementary form. Normalise the polynomial to be monic and let L be a finite extension of the base containing its coefficients, which exists because those coefficients are algebraic (*exists-finite-subfield-tower*).

Divide X^i by the polynomial for $i \leq n * \text{card } B$, where B is a basis of L : each remainder has degree below n and coefficients in L , so it is described by $n * \text{card } B$ scalars from the base field, while there are $n * \text{card } B + 1$ of them. *Subfield.underdetermined-solution* therefore supplies a nontrivial vanishing combination, whose coefficients assemble into a nonzero polynomial *over the base* that the given polynomial divides. That one splits in the closure, and a divisor of a split polynomial has a root.

Where HOL-Algebra forms the quotient $A[X]/(P)$ and compares its dimension with that of $K[X]$, the linear system does the same counting directly, so no quotient (which would change the type) is ever constructed.

```
theorem alg-closure-alg-closed:
  fixes  $p :: 'a :: \text{field wide poly}$ 
  assumes  $p: p \in \text{poly-over alg-closure}$  and  $dp: 0 < \text{degree } p$ 
  obtains  $z$  where  $z \in \text{alg-closure}$  and  $\text{poly } p \ z = 0$ 
  <proof>
```

end

84 Module homomorphisms

```
theory Module-Homomorphism
imports Module-Basics
begin
```

A *module homomorphism* (over a fixed ring R) is a map η between R -modules that is additive and commutes with scaling. This mirrors the group and ring homomorphism locales of theories *Group-Theory* and *Ring-Theory*, with the extra axiom linking the two scaling operations. We develop the kernel, image, and the injectivity criterion $\text{Ker} = \{0\}$.

The import of *FiniteProduct* (through *Module*) re-merges HOL's $+/-$ concrete syntax alongside the ring locale's, which is harmless in term positions (type inference disambiguates) but ambiguous in a locale-header instantiation argument. We therefore remove the HOL syntax again, as *Ring-Theory* does, so that within this theory $+$ unambiguously denotes the ring's addition.

```
no-notation plus (infixl  $\langle + \rangle$  65)
no-notation minus (infixl  $\langle - \rangle$  65)
unbundle no uminus-syntax
```

A module homomorphism fixes a *single* scalar ring R and two R -modules — the source M and the target M' — sharing that ring. The ring parameters $+$ $\cdot$ **0** **1** appear once in the *for* clause and are threaded verbatim into both module interpretations, so no token is bound twice.

```
locale module-homomorphism =
  source: Module  $R$   $(+)$   $(\cdot)$  0 1  $(\oplus)$  0 $M$   $M$   $(\odot)$   $+$ 
  target: Module  $R$   $(+)$   $(\cdot)$  0 1  $(\oplus_2)$  0 $2$   $M'$   $(\odot_2)$   $+$ 
```

```

map  $\eta$   $M$   $M'$ 
for  $R$  and addition (infixl  $\langle + \rangle$  65) and multiplication (infixl  $\langle \cdot \rangle$  70) and zero
( $\langle 0 \rangle$ ) and unit ( $\langle 1 \rangle$ )
  and  $M$  and madd (infixl  $\langle \oplus \rangle$  65) and mzero ( $\langle 0_M \rangle$ ) and scale (infixr  $\langle \odot \rangle$  75)
  and  $M'$  and madd' (infixl  $\langle \oplus_2 \rangle$  65) and mzero' ( $\langle 0_2 \rangle$ ) and scale' (infixr  $\langle \odot_2 \rangle$ 
75)
  and  $\eta$  +
  assumes hom-add:  $\llbracket u \in M; v \in M \rrbracket \implies \eta (u \oplus v) = \eta u \oplus_2 \eta v$ 
  and hom-scale:  $\llbracket a \in R; v \in M \rrbracket \implies \eta (a \odot v) = a \odot_2 \eta v$ 
begin

```

84.1 Basic properties

lemma *hom-closed*: $v \in M \implies \eta v \in M'$
 $\langle proof \rangle$

lemma *hom-zero*: $\eta 0_M = 0_2$
 $\langle proof \rangle$

lemma *hom-neg*:
 assumes $v: v \in M$ shows $\eta (source.madd.inverse v) = target.madd.inverse (\eta v)$
 $\langle proof \rangle$

A module homomorphism commutes with finite linear combinations. The target sum remains indexed by the source vectors; reindexing it over their images additionally requires injectivity on the chosen index set.

lemma *hom-lincomb*:
 assumes *finA*: *finite* A and $AM: A \subseteq M$
 and $c: \bigwedge v. v \in A \implies c v \in R$
 shows $\eta (source.lincomb c A) =$
 $target.madd.fincomp (\lambda v. c v \odot_2 \eta v) A$
 $\langle proof \rangle$

84.2 Composition and inverses

Composition is restricted to the source carrier, preserving the extensional-map convention used throughout the set-based algebra hierarchy.

theorem *compose-module-homomorphism*:
 assumes *second-hom*: *module-homomorphism* $R (+) (\cdot) 0 1$
 $M' (\oplus_2) 0_2 (\odot_2) N$ nadd nzero nscale ϑ
 shows *module-homomorphism* $R (+) (\cdot) 0 1$
 $M (\oplus) 0_M (\odot) N$ nadd nzero nscale $(compose M \vartheta \eta)$
 $\langle proof \rangle$

The inverse of a bijective homomorphism is again restricted to its source carrier. This is essential: a raw *inv-into* does not satisfy the repository's extensional-map convention away from the carrier.

theorem *inverse-module-homomorphism*:

assumes *bij*: *bij-betw* η M M'
shows *module-homomorphism* R $(+)$ $(\cdot)$ $\mathbf{0}$ $\mathbf{1}$
 $M' (\oplus_2) \mathbf{0}_2 (\odot_2)$
 $M (\oplus) \mathbf{0}_M (\odot) (\text{restrict } (\text{inv-into } M \ \eta) \ M')$
 $\langle \text{proof} \rangle$

lemma *inverse-module-homomorphism-left*:
assumes *bij*: *bij-betw* η M M' **and** $x: x \in M$
shows *restrict* $(\text{inv-into } M \ \eta) \ M' (\eta \ x) = x$
 $\langle \text{proof} \rangle$

lemma *inverse-module-homomorphism-right*:
assumes *bij*: *bij-betw* η M M' **and** $y: y \in M'$
shows $\eta (\text{restrict } (\text{inv-into } M \ \eta) \ M' \ y) = y$
 $\langle \text{proof} \rangle$

corollary *inverse-module-isomorphism*:
assumes *bij*: *bij-betw* η M M'
shows *module-homomorphism* R $(+)$ $(\cdot)$ $\mathbf{0}$ $\mathbf{1}$
 $M' (\oplus_2) \mathbf{0}_2 (\odot_2)$
 $M (\oplus) \mathbf{0}_M (\odot) (\text{restrict } (\text{inv-into } M \ \eta) \ M')$
 $\wedge \text{bij-betw } (\text{restrict } (\text{inv-into } M \ \eta) \ M') \ M' \ M$
 $\langle \text{proof} \rangle$

84.3 The kernel

definition *Ker* :: 'b set
where $Ker = \{v \in M. \ \eta \ v = \mathbf{0}_2\}$

lemma *Ker-subset*: $Ker \subseteq M$
 $\langle \text{proof} \rangle$

lemma *Ker-mem*: $v \in Ker \implies v \in M$
 $\langle \text{proof} \rangle$

lemma *Ker-image*: $v \in Ker \implies \eta \ v = \mathbf{0}_2$
 $\langle \text{proof} \rangle$

lemma *Ker-memI*: $\llbracket \eta \ v = \mathbf{0}_2; v \in M \rrbracket \implies v \in Ker$
 $\langle \text{proof} \rangle$

The kernel is a submodule of the source.

theorem *Ker-submodule*: *source.submodule* Ker
 $\langle \text{proof} \rangle$

84.4 The image

The image of η is a submodule of the target.

theorem *image-submodule*: *target.submodule* $(\eta \text{ ' } M)$

<proof>

84.5 Injectivity criterion

A module homomorphism is injective iff its kernel is trivial.

theorem *injective-iff-kernel-trivial:*

inj-on η $M \longleftrightarrow Ker = \{0_M\}$

<proof>

end

context *Module*

begin

The carrier-restricted identity is the identity morphism in the extensional-map representation.

theorem *identity-module-homomorphism:*

module-homomorphism R $(+)$ $(\cdot)$ 0 1

M $(\oplus)$ 0_M $(\odot)$ M $(\oplus)$ 0_M $(\odot)$ *(identity* $M)$

<proof>

end

end

85 Quotient modules and the first isomorphism theorem

theory *Quotient-Module*

imports *Module-Homomorphism*

begin

Given a submodule N of an R -module M , the quotient set M/N inherits a module structure with scaling $a \cdot Class(v) = Class(a \odot v)$. The construction reuses the group-quotient machinery of *Group-Theory* for the additive structure (any subgroup of an abelian group is normal, so cosets already form an abelian group under coset addition) and only requires the induced scaling to be shown well-defined.

On this base we prove the natural projection to be a module homomorphism with kernel N , and derive the first isomorphism theorem: for a module homomorphism η , the induced map on $M/Ker \eta$ is an isomorphism onto the image $\eta \text{ `` } M$.

85.1 The quotient module of a submodule

Work inside the *Module* locale, with a submodule fixed.

locale *submodule-in-module* = *Module* +

fixes $N :: 'b \text{ set}$
assumes $N\text{-submodule: submodule } N$
begin

The submodule is an additive subgroup of the module. Since the additive group is abelian, it is automatically normal.

lemma $N\text{-subgroup: Subgroup } N \ M \ (\oplus) \ \mathbf{0}_M$
 $\langle \text{proof} \rangle$

sublocale $madd\text{-sub: subgroup-of-abelian-group } N \ M \ (\oplus) \ \mathbf{0}_M$
 $\langle \text{proof} \rangle$

Notation: the quotient carrier, the coset class, and the induced coset addition.

abbreviation $Q\text{carrier} :: 'b \text{ set set } (\langle M' / N \rangle)$
where $M / N \equiv madd\text{-sub.Factor-Group}$

abbreviation $Q\text{class} :: 'b \Rightarrow 'b \text{ set}$
where $Q\text{class } v \equiv madd\text{-sub.Class } v$

notation $madd\text{-sub.quotient-composition}$ (**infixl** $\langle [\oplus] \rangle$ 65)

An explicit constant for the coset addition, so it can be referred to by name from outside the locale (in **sublocale**-free consumers).

abbreviation $qadd :: 'b \text{ set} \Rightarrow 'b \text{ set} \Rightarrow 'b \text{ set}$
where $qadd \equiv madd\text{-sub.quotient-composition}$

The additive coset structure is already a (commutative) group, inherited from *subgroup-of-abelian-group*.

lemma $qmodule\text{-additive: Abelian-Group } (M / N) \ ([\oplus]) \ (Q\text{class } \mathbf{0}_M)$
 $\langle \text{proof} \rangle$

85.1.1 The induced scaling is well-defined

Two module elements lie in the same coset iff their difference lies in N .

lemma $\text{Class-eq-iff-diff-in-}N$:
assumes $u: u \in M$ **and** $v: v \in M$
shows $Q\text{class } u = Q\text{class } v \longleftrightarrow u \oplus madd.inverse \ v \in N$
 $\langle \text{proof} \rangle$

The scaling is well-defined on cosets: if $u - v \in N$ (in the module additive sense) then $a \odot u - a \odot v \in N$, since $a \odot (u \oplus madd.inverse \ v)$ equals $(a \odot u) \oplus (a \odot madd.inverse \ v) = (a \odot u) \oplus madd.inverse \ (a \odot v)$ and lies in N by scaling-closedness of the submodule.

lemma $\text{scale-respects-Class}$:
assumes $a: a \in R$ **and** $uv: Q\text{class } u = Q\text{class } v$ **and** $u: u \in M$ **and** $v: v \in M$
shows $Q\text{class } (a \odot u) = Q\text{class } (a \odot v)$
 $\langle \text{proof} \rangle$

85.1.2 The quotient scaling operation and the quotient module

Choose any representative and scale it; well-definedness (*scale-respects-Class*) makes the choice immaterial.

definition $qscale :: 'a \Rightarrow 'b \text{ set} \Rightarrow 'b \text{ set}$ (**infixr** $\langle [\odot] \rangle$ 75)
where $a [\odot] X = (THE\ Y.\ \exists v \in M.\ X = Qclass\ v \wedge Y = Qclass\ (a \odot v))$

lemma *qscale-Class*:

assumes $a: a \in R$ **and** $v: v \in M$

shows $a [\odot] Qclass\ v = Qclass\ (a \odot v)$

<proof>

lemma *qscale-closed* [*intro, simp*]:

assumes $a: a \in R$ **and** $X: X \in M/N$ **shows** $a [\odot] X \in M/N$

<proof>

The six module axioms on the quotient, packaged as a *Module-axioms* fact. This is the pinch point: proving *Module-axioms* separately, and then composing with the ambient *Ring-axioms* via *Module.intro*, avoids the fragile mid-interpretation goal-order handshake we would face by opening a bare **sublocale** on *Module*.

lemma *quotient-module-axioms*:

Module-axioms $R\ (+)\ (\cdot)\ \mathbf{1}\ ([\oplus])\ (Qclass\ \mathbf{0}_M)\ (M/N)\ ([\odot])$

<proof>

sublocale *quotient*: *Module* $R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}\ ([\oplus])\ Qclass\ \mathbf{0}_M\ M/N\ ([\odot])$

<proof>

85.1.3 The natural projection

The natural projection $v \mapsto Qclass\ v$ extensional on M .

definition *nat-proj* :: $'b \Rightarrow 'b \text{ set}$

where $nat-proj = (\lambda v \in M.\ Qclass\ v)$

lemma *nat-proj-apply* [*simp*]: $v \in M \implies nat-proj\ v = Qclass\ v$

<proof>

lemma *nat-proj-undefined*: $v \notin M \implies nat-proj\ v = undefined$

<proof>

The natural projection is a module homomorphism with kernel N .

sublocale *nat-proj*: *module-homomorphism*

$R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$

$M\ (\oplus)\ \mathbf{0}_M\ (\odot)$

$M/N\ ([\oplus])\ Qclass\ \mathbf{0}_M\ ([\odot])$

nat-proj

<proof>

The natural projection maps onto the entire quotient carrier.

theorem *nat-proj-image*: $\text{nat-proj} \text{ ‘ } M = M/N$
 $\langle \text{proof} \rangle$

The kernel of the natural projection is exactly N .

theorem *nat-proj-Ker*: $\text{nat-proj.Ker} = N$
 $\langle \text{proof} \rangle$

end

end

86 The first isomorphism theorem for modules

theory *Module-First-Iso*
imports *Quotient-Module*
begin

Given a module homomorphism $\eta \in M \rightarrow M'$, its kernel is a submodule of M , and the induced map on the quotient $M/\text{Ker } \eta$ onto $\eta \text{ ‘ } M$ is a module isomorphism.

The natural way to organise this development inside *module-homomorphism* is to prove once and for all that the kernel gives rise to a *submodule-in-module* instance, and then use the operations $K.Qcarrier$, $K.Qclass$, $K.qadd$, $K.qscale$ supplied by that instance in the theorems below. We do this via **interpretation** K : *submodule-in-module* ... Ker at the head of the **context** block, discharged by the lemma *Ker-submodule-in-module* below.

context *module-homomorphism*
begin

86.1 The kernel-quotient and the induced map

The kernel is a submodule (*Ker-submodule*), so the *submodule-in-module* locale holds on it. This lemma is the witness fed to the **interpretation** below.

lemma *Ker-submodule-in-module*:
 $\text{submodule-in-module } R \text{ (+) } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1} \text{ } (\oplus) \text{ } \mathbf{0}_M \text{ } M \text{ } (\odot) \text{ } Ker$
 $\langle \text{proof} \rangle$

Install *submodule-in-module* on the kernel with prefix K . The theorems below then use $K.Qcarrier$, $K.Qclass$, $K.qadd$, $K.qscale$, and other facts derived from the locale ($K.Class\text{-eq-iff-diff-in-}N$, $K.qscale\text{-Class}$, $K.quotient.Module\text{-axioms}$, $K.madd\text{-sub.representant-exists}$, $K.madd\text{-sub.Class-commutes-with-composition}$).

interpretation K : *submodule-in-module*
 $R \text{ (+) } (\cdot) \text{ } \mathbf{0} \text{ } \mathbf{1} \text{ } (\oplus) \text{ } \mathbf{0}_M \text{ } M \text{ } (\odot) \text{ } Ker$
 $\langle \text{proof} \rangle$

The induced map on cosets. Made extensional on $K.Qcarrier$ so that it fits the *map* locale in the *module-homomorphism* obligation.

definition $ind :: 'b \text{ set} \Rightarrow 'c$
where $ind = (\lambda X \in K.Qcarrier. \text{THE } y. \exists v \in M. X = K.Qclass\ v \wedge y = \eta\ v)$

lemma *ind-Class*:
assumes $v: v \in M$ **shows** $ind\ (K.Qclass\ v) = \eta\ v$
 $\langle proof \rangle$

lemma *ind-closed*:
assumes $X \in K.Qcarrier$ **shows** $ind\ X \in \eta\ 'M$
 $\langle proof \rangle$

86.2 The image $\eta\ 'M$ is a submodule of the target

lemma *image-abelian-group*: *Abelian-Group* $(\eta\ 'M)\ (\oplus_2)\ \mathbf{0}_2$
 $\langle proof \rangle$

lemma *image-module-axioms*:
Module-axioms $R\ (+)\ (\cdot)\ \mathbf{1}\ (\oplus_2)\ \mathbf{0}_2\ (\eta\ 'M)\ (\odot_2)$
 $\langle proof \rangle$

lemma *image-is-module*: *Module* $R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}\ (\oplus_2)\ \mathbf{0}_2\ (\eta\ 'M)\ (\odot_2)$
 $\langle proof \rangle$

86.3 The induced map is a module homomorphism, injective, surjective

The quotient module on the kernel is a module (specialisation of the quotient's own *Module-axioms* to the kernel-as-submodule).

lemma *quotient-is-module*:
Module $R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}\ K.qadd\ (K.Qclass\ \mathbf{0}_M)\ K.Qcarrier\ K.qscale$
 $\langle proof \rangle$

theorem *ind-module-homomorphism*:
module-homomorphism $R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$
 $K.Qcarrier\ K.qadd\ (K.Qclass\ \mathbf{0}_M)\ K.qscale$
 $(\eta\ 'M)\ (\oplus_2)\ \mathbf{0}_2\ (\odot_2)\ ind$
 $\langle proof \rangle$

theorem *ind-inj*: *inj-on* $ind\ K.Qcarrier$
 $\langle proof \rangle$

theorem *ind-surj*: $ind\ 'K.Qcarrier = \eta\ 'M$
 $\langle proof \rangle$

The first isomorphism theorem: the induced map ind is a bijective module homomorphism from $M/Ker\ \eta$ onto $\eta\ 'M$.

theorem *first-isomorphism*:

```

module-homomorphism R (+) (·) 0 1
  K.Qcarrier K.qadd (K.Qclass 0_M) K.qscale
  (η ‘ M) (⊕2) 02 (⊙2) ind
  ∧ bij-betw ind K.Qcarrier (η ‘ M)
⟨proof⟩

```

end

end

87 The second and third isomorphism theorems for modules

The third isomorphism theorem for modules: for submodules $K \subseteq N \subseteq M$, $(M/K)/(N/K) \cong M/N$; and the second: for submodules N and K , $(N \oplus_S K)/K \cong N/(N \cap K)$, where $\oplus_S$ is the submodule sum defined in *Module*.

Performance. This theory loads in a fraction of a second. Earlier drafts of this comment claimed that the two proof-local **interpret**...: *submodule-in-module* ... calls per theorem were prohibitively expensive, and that this was why the theory was kept out of *ROOT*. That was a misdiagnosis: the long build times seen at the time were machine contention, not locale instantiation cost. There is nothing here to optimise.

```

theory Module-Iso-Theorems
  imports Module-First-Iso
begin

```

Building on the first isomorphism theorem in *Module-First-Iso*, we derive the third: given submodules $K \subseteq N \subseteq M$, the natural map $v + K \mapsto v + N$ is a surjective module homomorphism from M/K onto M/N with kernel N/K ; by the first iso, $(M/K) / (N/K) \cong M/N$.

*Note on the two **interprets**.* Each theorem below interprets *submodule-in-module* twice inside its own proof, once for K and once for N . This is cheap and needs no refactoring. A previous version of this note asserted that the alternative — two **interpretations** at the head of the enclosing **context** *Module* block — triggers a non-terminating locale roundup. That claim was never soundly established (it shares an origin with the contention misdiagnosis above) and should be re-measured before being repeated or acted upon.

```

context Module
begin

```

The natural map $v + K \mapsto v + N$ from M/K to M/N . Made extensional on M/K by construction so that the *map* obligation of *module-homomorphism* is directly discharged.

```

definition proj-KN :: 'b set ⇒ 'b set ⇒ 'b set ⇒ 'b set
  where proj-KN K N =

```

$(\lambda A \in \text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M K.$
 $\text{THE } B. \exists v \in M. A = \text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K v$
 $\wedge B = \text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M N v)$

Well-definedness of the projection: two representatives in the same K -coset lie in the same N -coset, because $K \subseteq N$. This is the workhorse fact for everything below, so we prove it once here rather than inside each theorem.

lemma *proj-KN-Class*:

assumes $N\text{-sub: submodule } N$ **and** $K\text{-sub: submodule } K$ **and** $K\text{-in-}N: K \subseteq N$ **and**
 $v: v \in M$

shows $\text{proj-KN } K N (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K v)$
 $= \text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M N v$

<proof>

The main theorem: for submodules K, N of M with $K \subseteq N$, the map $\text{proj-KN } K N$ is a surjective module homomorphism from M/K onto M/N whose kernel is $K.\text{Qclass } \text{' } N$ (which is N/K viewed inside M/K).

theorem *third-isomorphism-setup*:

assumes $N\text{-sub: submodule } N$

and $K\text{-sub: submodule } K$

and $K\text{-in-}N: K \subseteq N$

shows $\text{module-homomorphism } R (+) (\cdot) \mathbf{0} \mathbf{1}$

$(\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M K)$

$(\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M M K)$

$(\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K \mathbf{0}_M)$

$(\text{submodule-in-module.qscale } (\oplus) \mathbf{0}_M M (\odot) K)$

$(\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M N)$

$(\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M M N)$

$(\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M N \mathbf{0}_M)$

$(\text{submodule-in-module.qscale } (\oplus) \mathbf{0}_M M (\odot) N)$

$(\text{proj-KN } K N)$

$(\text{is module-homomorphism } - - - - ?QK ?qaddK ?zK ?qscaleK ?QN ?qaddN$

$?zN ?qscaleN -)$

<proof>

theorem *third-isomorphism-surj*:

assumes $N\text{-sub: submodule } N$

and $K\text{-sub: submodule } K$

and $K\text{-in-}N: K \subseteq N$

shows $\text{proj-KN } K N \text{' } (\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M K)$

$= \text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M N$

<proof>

87.1 The kernel of the projection is N/K

The kernel of $\text{proj-KN } K N$ consists of exactly those K -cosets whose representatives lie in N — that is, of N/K viewed as a subset of M/K . Note the kernel is computed with respect to the zero of M/N , namely $N.\text{Qclass } \mathbf{0}_M$.

theorem *phi-Ker*:

assumes *N-sub*: submodule *N*

and *K-sub*: submodule *K*

and *K-in-N*: $K \subseteq N$

shows $\{A \in \text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M \ M \ K.$
 $\text{proj-KN } K \ N \ A = \text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ N \ \mathbf{0}_M\}$
 $= \text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ ' \ N$

<proof>

87.2 The third isomorphism theorem

Assembling the pieces: *proj-KN* *K N* is a module homomorphism (*third-isomorphism-setup*) from M/K onto M/N (*third-isomorphism-surj*) with kernel N/K (*phi-Ker*). Applying the first isomorphism theorem to it yields $(M/K)/(N/K) \cong M/N$.

We state the conclusion as the *existence* of an isomorphism rather than by naming the induced map explicitly. Naming it would mean writing out *module-homomorphism.ind* applied to the iterated-quotient operations, which is a long and fragile parameter list; existence is the standard textbook phrasing and is what consumers actually use.

theorem *third-isomorphism*:

assumes *N-sub*: submodule *N*

and *K-sub*: submodule *K*

and *K-in-N*: $K \subseteq N$

shows $\exists \varphi. \text{ module-homomorphism } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$
 $(\text{submodule-in-module.Qcarrier}$
 $\ (\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ \mathbf{0}_M)$
 $\ (\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ ' \ N))$
 $(\text{submodule-in-module.qadd}$
 $\ (\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ \mathbf{0}_M)$
 $\ (\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ ' \ N))$
 $(\text{submodule-in-module.Qclass}$
 $\ (\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ \mathbf{0}_M)$
 $\ (\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ ' \ N)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ \mathbf{0}_M))$
 $(\text{submodule-in-module.qscale}$
 $\ (\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ \mathbf{0}_M)$
 $\ (\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M \ M \ K)$
 $\ (\text{submodule-in-module.qscale } (\oplus) \mathbf{0}_M \ M \ (\odot) \ K)$
 $\ (\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M \ M \ K \ ' \ N))$
 $(\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M \ M \ N)$
 $(\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M \ M \ N)$

$(\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M N \mathbf{0}_M)$
 $(\text{submodule-in-module.qscale } (\oplus) \mathbf{0}_M M (\odot) N)$
 φ
 $\wedge \text{bij-betw } \varphi$
 $(\text{submodule-in-module.Qcarrier}$
 $(\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M M K)$
 $(\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K \mathbf{0}_M)$
 $(\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M K)$
 $(\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K ' N))$
 $(\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M N)$
 $\langle \text{proof} \rangle$

87.3 The second isomorphism theorem

For submodules N and K , the second isomorphism theorem states $(N \oplus_S K)/K \cong N/(N \cap K)$.

The proof restricts the natural projection $M \rightarrow M/K$ to N . A submodule is itself a module, so the restriction is a module homomorphism from N to M/K ; its image is $(N \oplus_S K)/K$ and its kernel is $N \cap K$. The first isomorphism theorem applied to it then gives the result.

As with the third isomorphism theorem, the conclusion is stated as the *existence* of an isomorphism rather than by naming the induced map, whose argument list would otherwise have to be written out in full.

A submodule, viewed as a module in its own right.

lemma *submodule-is-module*:

assumes N : submodule N **shows** Module $R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus) \mathbf{0}_M N (\odot)$
 $\langle \text{proof} \rangle$

The natural projection $M \rightarrow M/K$ restricted to N .

definition *proj-res* :: 'b set $\Rightarrow$ 'b set $\Rightarrow$ 'b $\Rightarrow$ 'b set

where *proj-res* $N K = (\lambda v \in N. \text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K v)$

lemma *proj-res-apply* [*simp*]:

$v \in N \implies \text{proj-res } N K v = \text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K v$
 $\langle \text{proof} \rangle$

The restricted projection is a module homomorphism from N to M/K .

lemma *proj-res-hom*:

assumes N : submodule N **and** K : submodule K

shows module-homomorphism $R (+) (\cdot) \mathbf{0} \mathbf{1}$

$N (\oplus) \mathbf{0}_M (\odot)$

$(\text{submodule-in-module.Qcarrier } (\oplus) \mathbf{0}_M M K)$

$(\text{submodule-in-module.qadd } (\oplus) \mathbf{0}_M M K)$

$(\text{submodule-in-module.Qclass } (\oplus) \mathbf{0}_M M K \mathbf{0}_M)$

$(\text{submodule-in-module.qscale } (\oplus) \mathbf{0}_M M (\odot) K)$

$(\text{proj-res } N K)$

$\langle \text{proof} \rangle$

Its kernel is $N \cap K$: a coset $v + K$ with $v \in N$ is zero exactly when $v \in K$.

lemma *proj-res-Ker*:

assumes N : submodule N **and** K : submodule K

shows $\{v \in N. \text{proj-res } N \ K \ v = \text{submodule-in-module.Qclass } (\oplus) \ \mathbf{0}_M \ M \ K \ \mathbf{0}_M\} = N \cap K$
 $\langle \text{proof} \rangle$

Its image is $(N \oplus_S K)/K$: the K -cosets of elements of N are exactly the K -cosets of elements of $N \oplus_S K$, since adding an element of K to a representative does not change its coset.

lemma *proj-res-image*:

assumes N : submodule N **and** K : submodule K

shows $\text{proj-res } N \ K \ ' \ N$

$= \text{submodule-in-module.Qclass } (\oplus) \ \mathbf{0}_M \ M \ K \ ' (N \oplus_S K)$

$\langle \text{proof} \rangle$

If N complements K , the restricted natural projection identifies N with the whole quotient M/K . The two complement conditions have separate roles: trivial intersection gives injectivity through the kernel calculation, while the full submodule sum gives surjectivity through the image calculation.

theorem *complement-quotient-isomorphism*:

assumes N : submodule N **and** K : submodule K

and disjoint: $N \cap K = \{\mathbf{0}_M\}$

and full: $N \oplus_S K = M$

shows *module-homomorphism* $R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

$N \ (\oplus) \ \mathbf{0}_M \ (\odot)$

$(\text{submodule-in-module.Qcarrier } (\oplus) \ \mathbf{0}_M \ M \ K)$

$(\text{submodule-in-module.qadd } (\oplus) \ \mathbf{0}_M \ M \ K)$

$(\text{submodule-in-module.Qclass } (\oplus) \ \mathbf{0}_M \ M \ K \ \mathbf{0}_M)$

$(\text{submodule-in-module.qscale } (\oplus) \ \mathbf{0}_M \ M \ (\odot) \ K)$

$(\text{proj-res } N \ K)$

$\wedge \text{bij-betw } (\text{proj-res } N \ K) \ N$

$(\text{submodule-in-module.Qcarrier } (\oplus) \ \mathbf{0}_M \ M \ K)$

$\langle \text{proof} \rangle$

The second isomorphism theorem.

theorem *second-isomorphism*:

assumes N : submodule N **and** K : submodule K

shows $\exists \varphi. \text{module-homomorphism } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

$(\text{submodule-in-module.Qcarrier } (\oplus) \ \mathbf{0}_M \ N \ (N \cap K))$

$(\text{submodule-in-module.qadd } (\oplus) \ \mathbf{0}_M \ N \ (N \cap K))$

$(\text{submodule-in-module.Qclass } (\oplus) \ \mathbf{0}_M \ N \ (N \cap K) \ \mathbf{0}_M)$

$(\text{submodule-in-module.qscale } (\oplus) \ \mathbf{0}_M \ N \ (\odot) \ (N \cap K))$

$(\text{submodule-in-module.Qclass } (\oplus) \ \mathbf{0}_M \ M \ K \ ' (N \oplus_S K))$

$(\text{submodule-in-module.qadd } (\oplus) \ \mathbf{0}_M \ M \ K)$

$(\text{submodule-in-module.Qclass } (\oplus) \ \mathbf{0}_M \ M \ K \ \mathbf{0}_M)$

$(\text{submodule-in-module.qscale } (\oplus) \ \mathbf{0}_M \ M \ (\odot) \ K)$

φ

```

       $\wedge$  bij-betw  $\varphi$ 
      (submodule-in-module.Qcarrier ( $\oplus$ )  $\mathbf{0}_M$   $N$  ( $N \cap K$ ))
      (submodule-in-module.Qclass ( $\oplus$ )  $\mathbf{0}_M$   $M$   $K$  ' ( $N \oplus_S K$ ))
    <proof>

  end
end

```

88 Complementary submodules and projections

```

theory Module-Complements
  imports Module-Iso-Theorems
begin

```

An idempotent module endomorphism is a projection onto its image. The image and kernel therefore recover the two internal summands without any finite-dimensional assumption or choice of bases.

```

locale idempotent-module-endomorphism =
  module-homomorphism  $R$  (+) (·)  $\mathbf{0}$   $\mathbf{1}$ 
     $M$  ( $\oplus$ )  $\mathbf{0}_M$  ( $\odot$ )
     $M$  ( $\oplus$ )  $\mathbf{0}_M$  ( $\odot$ )  $p$ 
  for  $R$  and addition (infixl  $\langle + \rangle$  65) and multiplication (infixl  $\langle \cdot \rangle$  70)
    and zero ( $\langle \mathbf{0} \rangle$ ) and unit ( $\langle \mathbf{1} \rangle$ )
    and  $M$  and madd (infixl  $\langle \oplus \rangle$  65) and mzero ( $\langle \mathbf{0}_M \rangle$ )
    and scale (infixr  $\langle \odot \rangle$  75) and  $p$  +
  assumes idempotent:  $\bigwedge x. x \in M \implies p (p x) = p x$ 
begin

```

```

theorem image-kernel-complementary:
   $p \text{ ' } M \cap \text{Ker} = \{\mathbf{0}_M\}$ 
   $p \text{ ' } M \oplus_S \text{Ker} = M$ 
<proof>

```

```

end

```

```

context Module
begin

```

The projection onto N along K is obtained canonically from the quotient: first map an element of M to its class in M/K , then use the inverse of the restricted quotient isomorphism from N . Both maps are restricted to their carriers, so the resulting projection follows the extensional-map convention of the module hierarchy.

```

definition complement-projection :: 'b set  $\Rightarrow$  'b set  $\Rightarrow$  'b  $\Rightarrow$  'b
where complement-projection  $N$   $K$  =
  compose  $M$ 
    (restrict (inv-into  $N$  (proj-res  $N$   $K$ )))
    (submodule-in-module.Qcarrier ( $\oplus$ )  $\mathbf{0}_M$   $M$   $K$ ))

```


(submodule-in-module.nat-proj $(\oplus) \mathbf{0}_M M K$)

The construction is a module homomorphism whenever the two submodules are complementary. This packages the quotient projection, the complement-quotient isomorphism, and its carrier-restricted inverse.

theorem *complement-projection-hom:*
assumes *N: submodule N and K: submodule K*
and disjoint: $N \cap K = \{\mathbf{0}_M\}$
and full: $N \oplus_S K = M$
shows *module-homomorphism* $R (+) (\cdot) \mathbf{0} \mathbf{1}$
 $M (\oplus) \mathbf{0}_M (\odot) N (\oplus) \mathbf{0}_M (\odot)$
(complement-projection N K)
 $\langle \text{proof} \rangle$

The projection fixes the chosen summand.

lemma *complement-projection-apply-left:*
assumes *N: submodule N and K: submodule K*
and disjoint: $N \cap K = \{\mathbf{0}_M\}$
and full: $N \oplus_S K = M$
and $x: x \in N$
shows *complement-projection N K* $x = x$
 $\langle \text{proof} \rangle$

The complementary summand is annihilated.

lemma *complement-projection-apply-right:*
assumes *N: submodule N and K: submodule K*
and disjoint: $N \cap K = \{\mathbf{0}_M\}$
and full: $N \oplus_S K = M$
and $x: x \in K$
shows *complement-projection N K* $x = \mathbf{0}_M$
 $\langle \text{proof} \rangle$

On an explicitly decomposed element, the projection selects its left component.

lemma *complement-projection-apply-sum:*
assumes *N: submodule N and K: submodule K*
and disjoint: $N \cap K = \{\mathbf{0}_M\}$
and full: $N \oplus_S K = M$
and $u: u \in N$ **and** $v: v \in K$
shows *complement-projection N K* $(u \oplus v) = u$
 $\langle \text{proof} \rangle$

Every element of a full, disjoint submodule sum has exactly one ordered pair of components. Pairing the witnesses makes the uniqueness quantifier directly usable by clients.

theorem *complementary-decomposition-exists-unique:*
assumes *N: submodule N and K: submodule K*
and disjoint: $N \cap K = \{\mathbf{0}_M\}$
and full: $N \oplus_S K = M$

and $x: x \in M$
shows $\exists! uv. \text{fst } uv \in N \wedge \text{snd } uv \in K \wedge x = \text{fst } uv \oplus \text{snd } uv$
 $\langle \text{proof} \rangle$

The image is precisely the summand onto which the map projects.

theorem *complement-projection-image*:
assumes N : submodule N **and** K : submodule K
and *disjoint*: $N \cap K = \{\mathbf{0}_M\}$
and *full*: $N \oplus_S K = M$
shows *complement-projection* $N K \text{ ' } M = N$
 $\langle \text{proof} \rangle$

The kernel is precisely the complementary summand.

theorem *complement-projection-Ker*:
assumes N : submodule N **and** K : submodule K
and *disjoint*: $N \cap K = \{\mathbf{0}_M\}$
and *full*: $N \oplus_S K = M$
shows $\{x \in M. \text{complement-projection } N K x = \mathbf{0}_M\} = K$
 $\langle \text{proof} \rangle$

Applying a complementary projection twice has no further effect.

lemma *complement-projection-idempotent*:
assumes N : submodule N **and** K : submodule K
and *disjoint*: $N \cap K = \{\mathbf{0}_M\}$
and *full*: $N \oplus_S K = M$
and $x: x \in M$
shows *complement-projection* $N K (\text{complement-projection } N K x) =$
complement-projection $N K x$
 $\langle \text{proof} \rangle$

The two complementary projections recover the original element. This also cross-checks the orientation of the construction: the first map selects the N -component and the swapped map selects the K -component.

theorem *complement-projections-sum*:
assumes N : submodule N **and** K : submodule K
and *disjoint*: $N \cap K = \{\mathbf{0}_M\}$
and *full*: $N \oplus_S K = M$
and $x: x \in M$
shows *complement-projection* $N K x \oplus \text{complement-projection } K N x = x$
 $\langle \text{proof} \rangle$

end

end

89 Exact sequences of modules

theory *Module-Exact-Sequence*

```

imports Module-Complements
begin

locale composable-module-homomorphisms =
  first: module-homomorphism  $R (+) (\cdot) \mathbf{0} \mathbf{1}$ 
     $M (\oplus) \mathbf{0}_M (\odot) M' (\oplus_2) \mathbf{0}_2 (\odot_2) \eta +$ 
  second: module-homomorphism  $R (+) (\cdot) \mathbf{0} \mathbf{1}$ 
     $M' (\oplus_2) \mathbf{0}_2 (\odot_2) M'' (\oplus_3) \mathbf{0}_3 (\odot_3) \vartheta$ 
for  $R$  and addition (infixl  $\langle + \rangle$  65) and multiplication (infixl  $\langle \cdot \rangle$  70)
  and zero ( $\langle \mathbf{0} \rangle$ ) and unit ( $\langle \mathbf{1} \rangle$ )
  and  $M$  and madd (infixl  $\langle \oplus \rangle$  65) and mzero ( $\langle \mathbf{0}_M \rangle$ )
    and scale (infixr  $\langle \odot \rangle$  75)
  and  $M'$  and madd' (infixl  $\langle \oplus_2 \rangle$  65) and mzero' ( $\langle \mathbf{0}_2 \rangle$ )
    and scale' (infixr  $\langle \odot_2 \rangle$  75)
  and  $M''$  and madd'' (infixl  $\langle \oplus_3 \rangle$  65) and mzero'' ( $\langle \mathbf{0}_3 \rangle$ )
    and scale'' (infixr  $\langle \odot_3 \rangle$  75)
  and  $\eta$  and  $\vartheta$ 
begin

definition exact :: bool
  where exact  $\longleftrightarrow \eta \text{ ' } M = \text{second.Ker}$ 

definition short-exact :: bool
  where short-exact  $\longleftrightarrow \text{inj-on } \eta \text{ } M \wedge \text{exact} \wedge \vartheta \text{ ' } M' = M''$ 

definition right-split :: bool
  where right-split  $\longleftrightarrow$ 
    ( $\exists \sigma. \text{module-homomorphism } R (+) (\cdot) \mathbf{0} \mathbf{1}$ 
       $M'' (\oplus_3) \mathbf{0}_3 (\odot_3) M' (\oplus_2) \mathbf{0}_2 (\odot_2) \sigma \wedge$ 
      ( $\forall x \in M''. \vartheta (\sigma x) = x$ ))

definition left-split :: bool
  where left-split  $\longleftrightarrow$ 
    ( $\exists \varrho. \text{module-homomorphism } R (+) (\cdot) \mathbf{0} \mathbf{1}$ 
       $M' (\oplus_2) \mathbf{0}_2 (\odot_2) M (\oplus) \mathbf{0}_M (\odot) \varrho \wedge$ 
      ( $\forall x \in M. \varrho (\eta x) = x$ ))

lemma exact-iff: exact  $\longleftrightarrow \eta \text{ ' } M = \text{second.Ker}$ 
   $\langle \text{proof} \rangle$ 

lemma short-exact-iff:
  short-exact  $\longleftrightarrow \text{inj-on } \eta \text{ } M \wedge \eta \text{ ' } M = \text{second.Ker} \wedge \vartheta \text{ ' } M' = M''$ 
   $\langle \text{proof} \rangle$ 

lemma exact-imp-composition-zero:
  assumes ex: exact and  $x: x \in M$ 
  shows  $\vartheta (\eta x) = \mathbf{0}_3$ 
   $\langle \text{proof} \rangle$ 

lemma right-split-imp-surjective:

```

assumes *split: right-split*
shows $\vartheta \text{ ' } M' = M''$
 $\langle \text{proof} \rangle$

The image of an explicit right section is complementary to the kernel of the split map. The composite $\sigma \circ \vartheta$ is the corresponding idempotent projection on the middle module.

theorem *right-section-image-kernel-complementary:*

assumes *section-hom: module-homomorphism* $R (+) (\cdot) \mathbf{0} \mathbf{1}$
 $M'' (\oplus_3) \mathbf{0}_3 (\odot_3) M' (\oplus_2) \mathbf{0}_2 (\odot_2) \sigma$
and *sec-eq:* $\bigwedge x. x \in M'' \implies \vartheta (\sigma x) = x$
shows $\sigma \text{ ' } M'' \cap \text{second.Ker} = \{\mathbf{0}_2\}$
and *Module.submodule-sum* $(\oplus_2) (\sigma \text{ ' } M'') \text{second.Ker} = M'$
 $\langle \text{proof} \rangle$

theorem *right-split-obtains-image-kernel-complement:*

assumes *split: right-split*
obtains σ **where**
module-homomorphism $R (+) (\cdot) \mathbf{0} \mathbf{1}$
 $M'' (\oplus_3) \mathbf{0}_3 (\odot_3) M' (\oplus_2) \mathbf{0}_2 (\odot_2) \sigma$
 $\bigwedge x. x \in M'' \implies \vartheta (\sigma x) = x$
 $\sigma \text{ ' } M'' \cap \text{second.Ker} = \{\mathbf{0}_2\}$
Module.submodule-sum $(\oplus_2) (\sigma \text{ ' } M'') \text{second.Ker} = M'$
 $\langle \text{proof} \rangle$

theorem *short-exact-right-split-obtains-decomposition:*

assumes *short: short-exact* **and** *split: right-split*
obtains σ **where**
module-homomorphism $R (+) (\cdot) \mathbf{0} \mathbf{1}$
 $M'' (\oplus_3) \mathbf{0}_3 (\odot_3) M' (\oplus_2) \mathbf{0}_2 (\odot_2) \sigma$
 $\bigwedge x. x \in M'' \implies \vartheta (\sigma x) = x$
 $\sigma \text{ ' } M'' \cap \eta \text{ ' } M = \{\mathbf{0}_2\}$
Module.submodule-sum $(\oplus_2) (\sigma \text{ ' } M'') (\eta \text{ ' } M) = M'$
 $\langle \text{proof} \rangle$

A right section yields a retraction of the injection. The projection onto the image of η along the section image is followed by the inverse of η onto its image.

theorem *short-exact-right-split-imp-left-split:*

assumes *short: short-exact* **and** *split: right-split*
shows *left-split*
 $\langle \text{proof} \rangle$

A retraction of the injection makes its image a direct summand. Exactness then identifies the complementary summand with the quotient, so the second map has a carrier-restricted inverse that supplies a right section.

theorem *short-exact-left-split-imp-right-split:*

assumes *short: short-exact* **and** *split: left-split*
shows *right-split*
 $\langle \text{proof} \rangle$

theorem *short-exact-right-split-iff-left-split:*
assumes *short: short-exact*
shows *right-split $\longleftrightarrow$ left-split*
 $\langle proof \rangle$

end

context *Module*

begin

The carrier-restricted identity on a submodule is its inclusion into the ambient module.

theorem *submodule-inclusion-hom:*
assumes *N: submodule N*
shows *module-homomorphism R (+) ($\cdot$) $\mathbf{0}$ $\mathbf{1}$*
 $N (\oplus) \mathbf{0}_M (\odot) M (\oplus) \mathbf{0}_M (\odot) (\text{identity } N)$
 $\langle proof \rangle$

Complementary submodules give the canonical split short exact sequence $0 \longrightarrow K \longrightarrow M \longrightarrow N \longrightarrow 0$. The inclusion of N is a section of the complementary projection.

theorem *complementary-submodules-split-short-exact:*
assumes *N: submodule N and K: submodule K*
and disjoint: $N \cap K = \{\mathbf{0}_M\}$
and full: $N \oplus_S K = M$
shows *composable-module-homomorphisms.short-exact*
 $K M N \mathbf{0}_M (\text{identity } K) (\text{complement-projection } N K)$
 $\wedge$ *composable-module-homomorphisms.right-split*
 $R (+) ($\cdot$) $\mathbf{0}$ $\mathbf{1}$$

 $M (\oplus) \mathbf{0}_M (\odot) N (\oplus) \mathbf{0}_M (\odot)$
 $(\text{complement-projection } N K)$
 $\langle proof \rangle$

end

end

90 The direct product and direct sum of a family of modules

theory *Module-Family-Product*
imports *Module-Homomorphism*
begin

no-notation *plus* (**infixl** $\langle + \rangle$ 65)
no-notation *minus* (**infixl** $\langle - \rangle$ 65)
unbundle *no uminus-syntax*

The external direct product of a family of R -modules $Mod\ i\ (i \in I)$: the carrier is the extensional dependent function space $Pi_E\ I\ Mod$ with pointwise addition and scaling. As with the homomorphism locale, the scalar ring R is fixed *once* and shared by all the modules. We show the product is an R -module, the projections are module homomorphisms, and the direct *sum* (elements of finite support) is a submodule of the product.

90.1 The product locale

```

locale module-family =
  base: Ring  $R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}$ 
  for  $R$  and addition (infixl  $\langle + \rangle$  65) and multiplication (infixl  $\langle \cdot \rangle$  70) and zero
  ( $\langle \mathbf{0} \rangle$ ) and unit ( $\langle \mathbf{1} \rangle$ ) +
  fixes  $I :: 'i\ set$ 
    and  $Mod :: 'i \Rightarrow 'b\ set$ 
    and  $madd :: 'i \Rightarrow 'b \Rightarrow 'b \Rightarrow 'b$ 
    and  $mzero :: 'i \Rightarrow 'b$ 
    and  $scale :: 'i \Rightarrow 'a \Rightarrow 'b \Rightarrow 'b$ 
  assumes  $mod: \bigwedge i. i \in I \implies$ 
     $Module\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}\ (madd\ i)\ (mzero\ i)\ (Mod\ i)\ (scale\ i)$ 
begin

```

```

lemma comp-module:  $i \in I \implies Module\ R\ (+)\ (\cdot)\ \mathbf{0}\ \mathbf{1}\ (madd\ i)\ (mzero\ i)\ (Mod\ i)\$ 
   $(scale\ i)$ 
   $\langle proof \rangle$ 

```

90.2 Carrier and operations

```

definition Pcarrier ::  $('i \Rightarrow 'b)\ set$ 
  where  $Pcarrier = (\Pi_E\ i \in I. Mod\ i)$ 

definition Padd ::  $('i \Rightarrow 'b) \Rightarrow ('i \Rightarrow 'b) \Rightarrow ('i \Rightarrow 'b)$  (infixl  $\langle \oplus_\Pi \rangle$  65)
  where  $x \oplus_\Pi y = (\lambda i \in I. madd\ i\ (x\ i)\ (y\ i))$ 

definition Pzero ::  $'i \Rightarrow 'b\ (\langle \mathbf{0}_\Pi \rangle)$ 
  where  $\mathbf{0}_\Pi = (\lambda i \in I. mzero\ i)$ 

definition Pscale ::  $'a \Rightarrow ('i \Rightarrow 'b) \Rightarrow ('i \Rightarrow 'b)$  (infixr  $\langle \odot_\Pi \rangle$  75)
  where  $a \odot_\Pi x = (\lambda i \in I. scale\ i\ a\ (x\ i))$ 

lemma Pcarrier-memI:
   $\llbracket \bigwedge i. i \in I \implies x\ i \in Mod\ i; x \in extensional\ I \rrbracket \implies x \in Pcarrier$ 
   $\langle proof \rangle$ 

lemma Pcarrier-component:  $\llbracket x \in Pcarrier; i \in I \rrbracket \implies x\ i \in Mod\ i$ 
   $\langle proof \rangle$ 

lemma Pcarrier-extensional:  $x \in Pcarrier \implies x \in extensional\ I$ 
   $\langle proof \rangle$ 

```

lemma *Pcarrier-eqI*:

$\llbracket x \in Pcarrier; y \in Pcarrier; \bigwedge i. i \in I \implies x \ i = y \ i \rrbracket \implies x = y$
 $\langle proof \rangle$

lemma *Padd-apply*: $i \in I \implies (x \oplus_{\Pi} y) \ i = madd \ i \ (x \ i) \ (y \ i)$
 $\langle proof \rangle$

lemma *Pzero-apply*: $i \in I \implies \mathbf{0}_{\Pi} \ i = mzero \ i$
 $\langle proof \rangle$

lemma *Pscale-apply*: $i \in I \implies (a \odot_{\Pi} x) \ i = scale \ i \ a \ (x \ i)$
 $\langle proof \rangle$

lemma *Padd-closed* [*intro*, *simp*]:

assumes $x \in Pcarrier \ y \in Pcarrier$ **shows** $x \oplus_{\Pi} y \in Pcarrier$
 $\langle proof \rangle$

lemma *Pzero-closed* [*intro*, *simp*]: $\mathbf{0}_{\Pi} \in Pcarrier$
 $\langle proof \rangle$

lemma *Pscale-closed* [*intro*, *simp*]:

assumes $a \in R$ **and** $x \in Pcarrier$ **shows** $a \odot_{\Pi} x \in Pcarrier$
 $\langle proof \rangle$

90.3 The additive abelian group of the product

lemma *Padd-abelian-group*: *Abelian-Group* $Pcarrier \ (\oplus_{\Pi}) \ \mathbf{0}_{\Pi}$
 $\langle proof \rangle$

90.4 The product is a module

theorem *module-Pcarrier*: *Module* $R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \ (\oplus_{\Pi}) \ \mathbf{0}_{\Pi} \ Pcarrier \ (\odot_{\Pi})$
 $\langle proof \rangle$

sublocale *product*: *Module* $R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \ (\oplus_{\Pi}) \ Pzero \ Pcarrier \ (\odot_{\Pi})$
 $\langle proof \rangle$

90.5 Projections

definition *Pproj* :: $'i \Rightarrow ('i \Rightarrow 'b) \Rightarrow 'b$

where $Pproj \ i = restrict \ (\lambda x. x \ i) \ Pcarrier$

lemma *Pproj-apply*: $x \in Pcarrier \implies Pproj \ i \ x = x \ i$
 $\langle proof \rangle$

Each projection is additive and commutes with scaling (it is a module homomorphism onto the factor *Mod i*); we record the two defining equations.

lemma *Pproj-add*: $\llbracket i \in I; x \in Pcarrier; y \in Pcarrier \rrbracket \implies Pproj \ i \ (x \oplus_{\Pi} y) = madd \ i \ (Pproj \ i \ x) \ (Pproj \ i \ y)$

⟨proof⟩

lemma *Pproj-scale*: $\llbracket i \in I; a \in R; x \in Pcarrier \rrbracket \implies Pproj\ i\ (a \odot_{\Pi} x) = scale\ i\ a\ (Pproj\ i\ x)$
 ⟨proof⟩

90.6 The direct sum

The *direct sum* consists of the product elements of *finite support*: those x equal to the zero element in all but finitely many factors. It is a submodule of the product.

definition *Psum* :: ('i $\Rightarrow$ 'b) set
where $Psum = \{x \in Pcarrier. finite\ \{i \in I. x\ i \neq mzero\ i\}\}$

lemma *Psum-subset*: $Psum \subseteq Pcarrier$
 ⟨proof⟩

lemma *Psum-memI*:
 $\llbracket x \in Pcarrier; finite\ \{i \in I. x\ i \neq mzero\ i\} \rrbracket \implies x \in Psum$
 ⟨proof⟩

theorem *Psum-submodule*: *product.submodule Psum*
 ⟨proof⟩

end

end

91 The universal property of a free module

theory *Free-Module-Universal*
imports *Free-Module Module-Homomorphism*
begin

As in *Module-Homomorphism*: the import of *FiniteProduct* (through *Module*) re-merges HOL's $+/-$ concrete syntax alongside the ring locale's. That is harmless in term positions but ambiguous in the locale-header instantiation arguments below, so we remove the HOL syntax again.

no-notation *plus* (**infixl** $\langle + \rangle$ 65)
no-notation *minus* (**infixl** $\langle - \rangle$ 65)
unbundle *no uminus-syntax*

A free module is free in the sense of a universal property: a homomorphism out of it may be prescribed *arbitrarily* on a basis, and is then determined. Precisely: if B is a basis of the R -module M and f maps B into an R -module M' , then there is exactly one module homomorphism $M \rightarrow M'$ extending f .

This is what the uniqueness-of-coordinates results in *Free-Module* were for. Existence needs coordinates to be *readable* — every element has a representation

over a finite subset of the basis — and well-definedness needs them to be *unique* in the strong sense of *lin-indep-lincomb-unique-gen*, across different index sets, since nothing fixes an index set in advance.

The locale below fixes the ring once in the *for* clause and shares it between the two modules, exactly as *module-homomorphism* does; writing the two module interpretations with separate ring parameters would make the $+$ token ambiguous.

```

locale free-module-ext =
  source: Module R (+) (·) 0 1 ( $\oplus$ ) 0M M ( $\odot$ ) +
  target: Module R (+) (·) 0 1 ( $\oplus_2$ ) 02 M' ( $\odot_2$ )
  for R and addition (infixl  $\langle + \rangle$  65) and multiplication (infixl  $\langle \cdot \rangle$  70) and zero
  ( $\langle 0 \rangle$ ) and unit ( $\langle 1 \rangle$ )
    and M and madd (infixl  $\langle \oplus \rangle$  65) and mzero ( $\langle 0_M \rangle$ ) and scale (infixr  $\langle \odot \rangle$  75)
    and M' and madd' (infixl  $\langle \oplus_2 \rangle$  65) and mzero' ( $\langle 0_2 \rangle$ ) and scale' (infixr  $\langle \odot_2 \rangle$ 
  75)
    and B and f +
  assumes basis: source.module-basis B
    and f-closed:  $v \in B \implies f v \in M'$ 
begin

```

The basis lies in the source module, and is independent and spanning.

lemma *B-subset*: $B \subseteq M$
 $\langle \text{proof} \rangle$

lemma *B-lin-indep*: source.lin-indep B
 $\langle \text{proof} \rangle$

91.1 The twisted sum

The image of a linear combination must be $\bigoplus_{v \in A} c v \odot_2 f v$ — a sum in the *target* indexed by elements of the source. We give this “twisted” sum a name and prove for it the three facts the homomorphism obligations need: it is closed, it is unchanged by padding the index set with zero coefficients, and it is additive and homogeneous in the coefficients.

definition *tsum* :: ($'b \Rightarrow 'a \Rightarrow 'b \text{ set} \Rightarrow 'c$
where *tsum* $c A = \text{target.madd.fincomp } (\lambda v. c v \odot_2 f v) A$

lemma *tsum-Pi*:
assumes $A: A \subseteq B$ **and** $c: \text{source.coeffs-on } c A$
shows $(\lambda v. c v \odot_2 f v) \in A \rightarrow M'$
 $\langle \text{proof} \rangle$

lemma *tsum-closed* [*intro*, *simp*]:
assumes $A: A \subseteq B$ **and** $c: \text{source.coeffs-on } c A$
shows *tsum* $c A \in M'$
 $\langle \text{proof} \rangle$

lemma *tsum-empty* [simp]: $tsum\ c\ \{\} = \mathbf{0}_2$
 ⟨proof⟩

Only the values on the index set matter.

lemma *tsum-cong*:
assumes $A: A \subseteq B$ **and** $eq: \bigwedge v. v \in A \implies c\ v = d\ v$ **and** $d: source.coeffs-on\ d\ A$
shows $tsum\ c\ A = tsum\ d\ A$
 ⟨proof⟩

Padding the index set with zero coefficients does not change the sum: the extra terms are $\mathbf{0} \odot_2 f\ v = \mathbf{0}_2$.

lemma *tsum-mono-zero*:
assumes $finA': finite\ A'$ **and** $AA': A \subseteq A'$ **and** $A'B: A' \subseteq B$
and $zero: \bigwedge v. v \in A' \setminus A \implies c\ v = \mathbf{0}$ **and** $c: source.coeffs-on\ c\ A'$
shows $tsum\ c\ A = tsum\ c\ A'$
 ⟨proof⟩

Additivity in the coefficients.

lemma *tsum-add-coeffs*:
assumes $A: A \subseteq B$ **and** $c: source.coeffs-on\ c\ A$ **and** $d: source.coeffs-on\ d\ A$
shows $tsum\ (\lambda v. c\ v + d\ v)\ A = tsum\ c\ A \oplus_2 tsum\ d\ A$
 ⟨proof⟩

Homogeneity in the coefficients.

lemma *tsum-scale-coeffs*:
assumes $A: A \subseteq B$ **and** $a: a \in R$ **and** $c: source.coeffs-on\ c\ A$
shows $tsum\ (\lambda v. a \cdot c\ v)\ A = a \odot_2 tsum\ c\ A$
 ⟨proof⟩

91.2 The extension is well defined

Two representations of the same element give the same twisted sum. This is where *lin-indep-lincomb-unique-gen* is used: the coefficient functions, extended by $\mathbf{0}$, agree, and each twisted sum equals the one over the union of the two index sets.

lemma *tsum-well-defined*:
assumes $A: finite\ A\ A \subseteq B$ **and** $A': finite\ A'\ A' \subseteq B$
and $c: source.coeffs-on\ c\ A$ **and** $d: source.coeffs-on\ d\ A'$
and $eq: source.lincomb\ c\ A = source.lincomb\ d\ A'$
shows $tsum\ c\ A = tsum\ d\ A'$
 ⟨proof⟩

91.3 The extension

The extension of f to all of M : read off any representation over the basis and take the twisted sum. Made extensional on M so that it satisfies the *map* obligation of *module-homomorphism* directly.

definition *free-ext* :: 'b $\Rightarrow$ 'c
where *free-ext* =
 $(\lambda x \in M. \text{THE } y. \exists A \ c. \text{finite } A \wedge A \subseteq B \wedge \text{source.coeffs-on } c \ A$
 $\wedge x = \text{source.lincomb } c \ A \wedge y = \text{tsum } c \ A)$

On a representation, the extension is the twisted sum.

lemma *free-ext-repr*:
assumes *A*: *finite A* $A \subseteq B$ **and** *c*: *source.coeffs-on c A* **and** *x*: $x = \text{source.lincomb } c \ A$
shows *free-ext x = tsum c A*
 $\langle \text{proof} \rangle$

The extension agrees with *f* on the basis: a basis element is its own singleton combination with coefficient **1**.

theorem *free-ext-extends*:
assumes *v*: $v \in B$ **shows** *free-ext v = f v*
 $\langle \text{proof} \rangle$

Existence. The extension is a module homomorphism.

theorem *free-ext-hom*:
module-homomorphism $R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \ M \ (\oplus) \ \mathbf{0}_M \ (\odot) \ M' \ (\oplus_2) \ \mathbf{0}_2 \ (\odot_2) \ \text{free-ext}$
 $\langle \text{proof} \rangle$

Uniqueness. Any homomorphism agreeing with *f* on the basis equals the extension: two homomorphisms agreeing on a spanning set agree everywhere, because each element is a finite combination of basis elements and both maps commute with the combination.

theorem *free-ext-unique*:
assumes η : *module-homomorphism* $R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \ M \ (\oplus) \ \mathbf{0}_M \ (\odot) \ M' \ (\oplus_2) \ \mathbf{0}_2 \ (\odot_2)$
 η
and *agree*: $\bigwedge v. v \in B \implies \eta \ v = f \ v$
and *x*: $x \in M$
shows $\eta \ x = \text{free-ext } x$
 $\langle \text{proof} \rangle$

The universal property. There is exactly one homomorphism extending *f*.

theorem *free-module-universal*:
 $\exists ! \eta. \text{module-homomorphism } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1} \ M \ (\oplus) \ \mathbf{0}_M \ (\odot) \ M' \ (\oplus_2) \ \mathbf{0}_2 \ (\odot_2) \ \eta$
 $\wedge (\forall v \in B. \eta \ v = f \ v) \wedge \eta \in M \rightarrow_E M'$
 $\langle \text{proof} \rangle$

end

end

92 The rank–nullity theorem

theory *Rank-Nullity*

imports *Steinitz Module-Homomorphism*
begin

As in the module-homomorphism theory, suppress HOL arithmetic syntax while binding the scalar-ring operations in the locale header.

no-notation *plus* (**infixl** $\langle + \rangle$ 65)
no-notation *minus* (**infixl** $\langle - \rangle$ 65)
unbundle *no uminus-syntax*

A linear map is a module homomorphism between vector spaces over the same scalar field.

locale *linear-map* =
 source: *Vector-Space* $R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus) \mathbf{0}_M M (\odot) +$
 target: *Vector-Space* $R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus_2) \mathbf{0}_2 M' (\odot_2) +$
 hom: *module-homomorphism*
 $R (+) (\cdot) \mathbf{0} \mathbf{1}$
 $M (\oplus) \mathbf{0}_M (\odot)$
 $M' (\oplus_2) \mathbf{0}_2 (\odot_2) \eta$
for R **and** *addition* (**infixl** $\langle + \rangle$ 65) **and** *multiplication* (**infixl** $\langle \cdot \rangle$ 70)
and *zero* ($\langle \mathbf{0} \rangle$) **and** *unit* ($\langle \mathbf{1} \rangle$)
and M **and** *madd* (**infixl** $\langle \oplus \rangle$ 65) **and** *mzero* ($\langle \mathbf{0}_M \rangle$)
and *scale* (**infixr** $\langle \odot \rangle$ 75)
and M' **and** *madd'* (**infixl** $\langle \oplus_2 \rangle$ 65) **and** *mzero'* ($\langle \mathbf{0}_2 \rangle$)
and *scale'* (**infixr** $\langle \odot_2 \rangle$ 75) **and** η
begin

The kernel and image inherit vector-space structures from the source and target.

sublocale *kernel: vector-subspace*
 $R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus) \mathbf{0}_M M (\odot) \text{hom.Ker}$
 $\langle \text{proof} \rangle$

sublocale *image: vector-subspace*
 $R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus_2) \mathbf{0}_2 M' (\odot_2) \eta \text{' } M$
 $\langle \text{proof} \rangle$

Reindex a target linear combination over the images of a finite set. Injectivity is exactly what makes the transported coefficient function unambiguous.

lemma *image-lincomb-reindex*:
assumes *finD*: *finite* D **and** *DM*: $D \subseteq M$ **and** *inj*: *inj-on* η D
and c : $\bigwedge v. v \in D \implies c \ v \in R$
defines $d \equiv \lambda y. c \ (\text{inv-into } D \ \eta \ y)$
shows *image.sub.lincomb* $d \ (\eta \text{' } D) =$
 $\text{target.vadd.fincomp } (\lambda v. c \ v \odot_2 \ \eta \ v) \ D$
 $\langle \text{proof} \rangle$

Once a kernel basis has been extended to a source basis, the map is injective on the complementary basis vectors. Equality of two images puts their difference in the kernel; the kernel basis would then express either vector in the span of the remaining source basis.

lemma *injective-on-basis-complement*:
assumes A : *kernel.sub.basis* A **and** C : *source.basis* C **and** AC : $A \subseteq C$
shows *inj-on* η $(C \setminus A)$
 $\langle$ *proof* $\rangle$

Split a linear combination over a basis into a chosen subset and its complement.

lemma *source-lincomb-diff*:
assumes *finC*: *finite* C **and** CM : $C \subseteq M$ **and** AC : $A \subseteq C$
and c : $\bigwedge v. v \in C \implies c \ v \in R$
shows *source.lincomb* c $C =$
source.lincomb c $A \oplus$ *source.lincomb* c $(C \setminus A)$
 $\langle$ *proof* $\rangle$

The images of the complementary basis vectors form a basis of the image.

lemma *image-basis-complement*:
assumes A : *kernel.sub.basis* A **and** C : *source.basis* C **and** AC : $A \subseteq C$
defines $D \equiv C \setminus A$
shows *image.sub.basis* $(\eta \restriction D)$
 $\langle$ *proof* $\rangle$

Rank–nullity follows by extending a basis of the kernel to a basis of the source.
The complementary vectors map bijectively to a basis of the image.

theorem *rank-nullity*:
assumes B : *source.basis* B
shows *plus kernel.sub.dimension image.sub.dimension = source.dimension*
 $\langle$ *proof* $\rangle$

A bijective linear map preserves dimension. Its kernel is the trivial subspace, so the image of any source basis is a basis of the entire target.

theorem *dimension-eq-of-bij-betw*:
assumes B : *source.basis* B **and** *bij*: *bij-betw* η M M'
shows *source.dimension = target.dimension*
 $\langle$ *proof* $\rangle$

92.1 Finite-dimensional consequences

Rank–nullity immediately bounds both the kernel and the image by the source dimension. These named forms avoid making clients repeat arithmetic reasoning about the decomposition.

corollary *kernel-dimension-le*:
assumes B : *source.basis* B
shows *kernel.sub.dimension* $\leq$ *source.dimension*
 $\langle$ *proof* $\rangle$

corollary *image-dimension-le*:
assumes B : *source.basis* B
shows *image.sub.dimension* $\leq$ *source.dimension*

⟨proof⟩

An injective map has trivial kernel, so rank-nullity identifies the source dimension with the image dimension.

theorem *image-dimension-eq-of-injective:*
 assumes *B: source.basis B and inj: inj-on η M*
 shows *image.sub.dimension = source.dimension*
⟨proof⟩

Comparing the image of an injective map with the finite-dimensional target gives the expected dimension inequality.

theorem *source-dimension-le-of-injective:*
 assumes *B: source.basis B and C: target.basis C and inj: inj-on η M*
 shows *source.dimension $\leq$ target.dimension*
⟨proof⟩

A surjective map has the whole target as its image, so the image bound from rank-nullity compares the target dimension directly with the source dimension.

theorem *target-dimension-le-of-surjective:*
 assumes *B: source.basis B and surj: $\eta \text{ ' } M = M'$*
 shows *target.dimension $\leq$ source.dimension*
⟨proof⟩

For finite-dimensional spaces of equal dimension, a linear map is injective exactly when it is surjective. Injectivity gives an image of full target dimension; conversely, surjectivity and rank-nullity force the kernel to have dimension zero.

theorem *injective-iff-surjective:*
 assumes *B: source.basis B and C: target.basis C*
 and *dim: source.dimension = target.dimension*
 shows *inj-on η M $\longleftrightarrow$ $\eta \text{ ' } M = M'$*
⟨proof⟩

end

notation *plus* (infixl $\langle + \rangle$ 65)
notation *minus* (infixl $\langle - \rangle$ 65)
unbundle *uminus-syntax*

end

93 Isomorphisms of finite-dimensional vector spaces

theory *Finite-Dimensional-Isomorphism*
 imports *Rank-Nullity Free-Module-Universal*
begin

Suppress HOL arithmetic syntax while binding the scalar-ring operations in the locale header, as in the other module-homomorphism theories.

no-notation *plus* (**infixl** $\langle + \rangle$ 65)
no-notation *minus* (**infixl** $\langle - \rangle$ 65)
unbundle *no uminus-syntax*

A pair of vector spaces over the same scalar field is the ambient setting for constructing a linear isomorphism before a particular map has been chosen.

locale *paired-vector-spaces* =
source: *Vector-Space* R $(+)$ $(\cdot)$ **0** **1** $(\oplus)$ $\mathbf{0}_M$ M $(\odot)$ $+$
target: *Vector-Space* R $(+)$ $(\cdot)$ **0** **1** $(\oplus_2)$ $\mathbf{0}_2$ M' $(\odot_2)$
for R **and** *addition* (**infixl** $\langle + \rangle$ 65) **and** *multiplication* (**infixl** $\langle \cdot \rangle$ 70)
and *zero* $(\langle \mathbf{0} \rangle)$ **and** *unit* $(\langle \mathbf{1} \rangle)$
and M **and** *madd* (**infixl** $\langle \oplus \rangle$ 65) **and** *mzero* $(\langle \mathbf{0}_M \rangle)$
and *scale* (**infixr** $\langle \odot \rangle$ 75)
and M' **and** *madd'* (**infixl** $\langle \oplus_2 \rangle$ 65) **and** *mzero'* $(\langle \mathbf{0}_2 \rangle)$
and *scale'* (**infixr** $\langle \odot_2 \rangle$ 75)
begin

A bijection between finite bases extends uniquely to a module homomorphism. Its image contains the target basis and hence, by minimality of span, the whole target. Equal basis cardinalities then let rank-nullity turn this surjection into a bijection.

theorem *basis-isomorphism-exists*:
assumes B : *source.basis* B **and** C : *target.basis* C **and** *card-eq*: *card* B = *card* C
shows $\exists \eta$.
module-homomorphism R $(+)$ $(\cdot)$ **0** **1**
 M $(\oplus)$ $\mathbf{0}_M$ $(\odot)$ M' $(\oplus_2)$ $\mathbf{0}_2$ $(\odot_2)$ η
 $\wedge$ *bij-betw* η M M' $\wedge$ *bij-betw* η B C
 $\langle \text{proof} \rangle$

Thus any chosen bases of equal-dimensional finite vector spaces can be matched by a linear isomorphism.

corollary *isomorphism-exists-of-equal-dimension*:
assumes B : *source.basis* B **and** C : *target.basis* C
and *dim*: *source.dimension* = *target.dimension*
shows $\exists \eta$.
module-homomorphism R $(+)$ $(\cdot)$ **0** **1**
 M $(\oplus)$ $\mathbf{0}_M$ $(\odot)$ M' $(\oplus_2)$ $\mathbf{0}_2$ $(\odot_2)$ η
 $\wedge$ *bij-betw* η M M' $\wedge$ *bij-betw* η B C
 $\langle \text{proof} \rangle$

Combining existence with invariance of dimension gives the classification theorem: two finite-dimensional vector spaces over the same field are linearly isomorphic exactly when their dimensions agree.

theorem *isomorphism-exists-iff-dimension-eq*:
assumes B : *source.basis* B **and** C : *target.basis* C
shows $(\exists \eta$.
module-homomorphism R $(+)$ $(\cdot)$ **0** **1**
 M $(\oplus)$ $\mathbf{0}_M$ $(\odot)$ M' $(\oplus_2)$ $\mathbf{0}_2$ $(\odot_2)$ η

```

       $\wedge$  bij-betw  $\eta$   $M$   $M'$ )
     $\longleftrightarrow$  source.dimension = target.dimension
  <proof>

```

end

```

notation plus (infixl  $\langle + \rangle$  65)
notation minus (infixl  $\langle - \rangle$  65)
unbundle uminus-syntax

```

end

94 Complements in finite-dimensional vector spaces

```

theory Finite-Dimensional-Splitting
  imports Rank-Nullity Quotient-Module Module-Iso-Theorems
begin

```

94.1 Dimension of quotient spaces

```

no-notation plus (infixl  $\langle + \rangle$  65)
no-notation minus (infixl  $\langle - \rangle$  65)
unbundle no uminus-syntax

```

A quotient of a vector space by a subspace inherits the vector-space structure already constructed at the module level. The natural projection is therefore a linear map.

```

locale vector-quotient = vector-subspace
begin

```

```

sublocale factor: submodule-in-module
   $R$   $(+)$   $(\cdot)$   $\mathbf{0}$   $\mathbf{1}$   $(\oplus)$   $\mathbf{0}_V$   $V$   $(\odot)$   $W$ 
  <proof>

```

```

lemma quotient-vector-space:
  Vector-Space  $R$   $(+)$   $(\cdot)$   $\mathbf{0}$   $\mathbf{1}$ 
    factor.qadd (factor.Qclass  $\mathbf{0}_V$ ) factor.Qcarrier factor.qscale
  <proof>

```

```

sublocale quotient: Vector-Space
   $R$   $(+)$   $(\cdot)$   $\mathbf{0}$   $\mathbf{1}$ 
    factor.qadd factor.Qclass  $\mathbf{0}_V$  factor.Qcarrier factor.qscale
  <proof>

```

```

sublocale projection: linear-map
   $R$   $(+)$   $(\cdot)$   $\mathbf{0}$   $\mathbf{1}$ 
   $V$   $(\oplus)$   $\mathbf{0}_V$   $(\odot)$ 
    factor.Qcarrier factor.qadd factor.Qclass  $\mathbf{0}_V$  factor.qscale factor.nat-proj
  <proof>

```


The dimension of a subspace plus the dimension of its quotient is the dimension of the ambient space. This is rank–nullity for the natural quotient projection.

theorem *quotient-dimension:*
assumes *B: basis B*
shows *plus sub.dimension quotient.dimension = dimension*
<proof>

lemma *quotient-basis-exists:*
assumes *B: basis B*
shows $\exists C. \text{quotient.basis } C$
<proof>

end

94.2 Grassmann’s dimension formula

locale *vector-subspace-pair* = *Vector-Space* +
fixes *U W :: 'b set*
assumes *U-submodule: mod.submodule U*
and *W-submodule: mod.submodule W*
begin

abbreviation *sum-carrier* **where**
sum-carrier $\equiv \text{mod.submodule-sum } U \ W$

abbreviation *restricted-projection* **where**
restricted-projection $\equiv \text{mod.proj-res } U \ W$

abbreviation *ambient-submodule* **where**
ambient-submodule $\equiv \text{mod.submodule}$

lemmas *ambient-submodule-inter* = *mod.submodule-inter*
lemmas *ambient-sum-submodule* = *mod.submodule-sum-submodule*
lemmas *ambient-sum-incl-right* = *mod.submodule-sum-incl-right*
lemmas *ambient-submodule-subset* = *mod.submodule-subset*
lemmas *ambient-submodule-zero* = *mod.submodule-zero*
lemmas *ambient-submodule-add* = *mod.submodule-add*
lemmas *ambient-submodule-scale* = *mod.submodule-scale*
lemmas *ambient-submodule-vector-space* = *submodule-vector-space*
lemmas *ambient-proj-res-hom* = *mod.proj-res-hom*
lemmas *ambient-proj-res-Ker* = *mod.proj-res-Ker*
lemmas *ambient-proj-res-image* = *mod.proj-res-image*

sublocale *left: vector-subspace*
R (+) (·) 0 1 (⊕) 0_V V (⊙) U
<proof>

sublocale *right: vector-quotient*
R (+) (·) 0 1 (⊕) 0_V V (⊙) W

$\langle proof \rangle$

sublocale *inter: vector-subspace*

$R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus) \mathbf{0}_V V (\odot) U \cap W$

$\langle proof \rangle$

sublocale *sum: vector-subspace*

$R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus) \mathbf{0}_V V (\odot) \text{sum-carrier}$

$\langle proof \rangle$

The right summand is also a subspace of the sum.

sublocale *sum-quotient: vector-quotient*

$R (+) (\cdot) \mathbf{0} \mathbf{1} (\oplus) \mathbf{0}_V \text{sum-carrier} (\odot) W$

$\langle proof \rangle$

Restricting the quotient projection by W to U gives a linear map whose kernel is $U \cap W$ and whose image consists of the classes represented in the sum.

lemma *restricted-projection-hom:*

module-homomorphism $R (+) (\cdot) \mathbf{0} \mathbf{1}$

$U (\oplus) \mathbf{0}_V (\odot)$

$\text{right.factor.Qcarrier right.factor.qadd (right.factor.Qclass } \mathbf{0}_V)$

$\text{right.factor.qscale restricted-projection}$

$\langle proof \rangle$

lemma *restricted-linear-map:*

linear-map $R (+) (\cdot) \mathbf{0} \mathbf{1}$

$U (\oplus) \mathbf{0}_V (\odot)$

$\text{right.factor.Qcarrier right.factor.qadd (right.factor.Qclass } \mathbf{0}_V)$

$\text{right.factor.qscale restricted-projection}$

$\langle proof \rangle$

sublocale *restricted: linear-map*

$R (+) (\cdot) \mathbf{0} \mathbf{1}$

$U (\oplus) \mathbf{0}_V (\odot)$

$\text{right.factor.Qcarrier right.factor.qadd right.factor.Qclass } \mathbf{0}_V$

$\text{right.factor.qscale restricted-projection}$

$\langle proof \rangle$

lemma *quotient-class-eq:*

assumes $x: x \in \text{sum-carrier}$

shows $\text{sum-quotient.factor.Qclass } x = \text{right.factor.Qclass } x$

$\langle proof \rangle$

lemma *sum-quotient-carrier-eq:*

$\text{sum-quotient.factor.Qcarrier} = \text{right.factor.Qclass } ' \text{sum-carrier}$

$\langle proof \rangle$

lemma *sum-quotient-carrier-eq-image:*

$\text{sum-quotient.factor.Qcarrier} = \text{restricted-projection } ' U$

<proof>

lemma *quotient-add-eq*:

assumes $X: X \in \text{sum-quotient.factor.Qcarrier}$

and $Y: Y \in \text{sum-quotient.factor.Qcarrier}$

shows $\text{sum-quotient.factor.qadd } X \ Y = \text{right.factor.qadd } X \ Y$

<proof>

lemma *quotient-scale-eq*:

assumes $a: a \in R$ **and** $X: X \in \text{sum-quotient.factor.Qcarrier}$

shows $\text{sum-quotient.factor.qscale } a \ X = \text{right.factor.qscale } a \ X$

<proof>

definition *quotient-embedding* **where**

$\text{quotient-embedding} = \text{restrict id sum-quotient.factor.Qcarrier}$

lemma *quotient-embedding-apply* [simp]:

$X \in \text{sum-quotient.factor.Qcarrier} \implies \text{quotient-embedding } X = X$

<proof>

lemma *quotient-embedding-hom*:

$\text{module-homomorphism } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

$\text{sum-quotient.factor.Qcarrier sum-quotient.factor.qadd}$

$(\text{sum-quotient.factor.Qclass } \mathbf{0}_V) \ \text{sum-quotient.factor.qscale}$

$(\text{restricted-projection } 'U) \ \text{right.factor.qadd } (\text{right.factor.Qclass } \mathbf{0}_V)$

$\text{right.factor.qscale quotient-embedding}$

<proof>

lemma *quotient-embedding-linear-map*:

$\text{linear-map } R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

$\text{sum-quotient.factor.Qcarrier sum-quotient.factor.qadd}$

$(\text{sum-quotient.factor.Qclass } \mathbf{0}_V) \ \text{sum-quotient.factor.qscale}$

$(\text{restricted-projection } 'U) \ \text{right.factor.qadd } (\text{right.factor.Qclass } \mathbf{0}_V)$

$\text{right.factor.qscale quotient-embedding}$

<proof>

sublocale *quotient-embedding: linear-map*

$R \ (+) \ (\cdot) \ \mathbf{0} \ \mathbf{1}$

$\text{sum-quotient.factor.Qcarrier sum-quotient.factor.qadd}$

$\text{sum-quotient.factor.Qclass } \mathbf{0}_V \ \text{sum-quotient.factor.qscale}$

$\text{restricted-projection } 'U \ \text{right.factor.qadd right.factor.Qclass } \mathbf{0}_V$

$\text{right.factor.qscale quotient-embedding}$

<proof>

lemma *quotient-embedding-bij*:

$\text{bij-betw quotient-embedding sum-quotient.factor.Qcarrier } (\text{restricted-projection } 'U)$

<proof>

Grassmann's dimension formula. The intersection dimension plus the sum dimension equals the sum of the two subspace dimensions.

theorem *grassmann-dimension:*

assumes *B: basis B*

shows *plus inter.sub.dimension sum.sub.dimension =
plus left.sub.dimension right.sub.dimension*

<proof>

corollary *direct-sum-dimension:*

assumes *B: basis B and disjoint: $U \cap W = \{0_V\}$*

shows *sum.sub.dimension = plus left.sub.dimension right.sub.dimension*

<proof>

end

94.3 Complements in finite-dimensional vector spaces

context *vector-subspace*

begin

Every subspace of a finite-dimensional vector space has a complementary subspace. Extend a basis of the subspace to an ambient basis and take the span of the added basis vectors. Independence gives the trivial intersection, while span minimality gives the full sum.

theorem *complementary-subspace-exists:*

assumes *B: basis B*

shows *$\exists U. \text{mod.submodule } U \wedge W \cap U = \{0_V\} \wedge$
 $\text{mod.submodule-sum } W U = V$*

<proof>

end

context *vector-quotient*

begin

Consequently every quotient of a finite-dimensional vector space is represented by a complementary subspace of the ambient space. The isomorphism is not merely existential: it is the natural quotient projection restricted to that complement.

theorem *complementary-quotient-isomorphism-exists:*

assumes *B: basis B*

shows *$\exists U. \text{mod.submodule } U$*

$\wedge W \cap U = \{0_V\}$

$\wedge \text{mod.submodule-sum } W U = V$

$\wedge \text{module-homomorphism } R (+) (\cdot) 0 1$

$U (\oplus) 0_V (\odot)$

$\text{factor.Qcarrier factor.qadd (factor.Qclass } 0_V) \text{ factor.qscale}$

$(\text{mod.proj-res } U W)$

$\wedge \text{bij-betw } (\text{mod.proj-res } U W) U \text{ factor.Qcarrier}$

<proof>

end

notation *plus* (**infixl** $\langle + \rangle$ 65)
notation *minus* (**infixl** $\langle - \rangle$ 65)
unbundle *uminus-syntax*

end

95 Simple Algebraic Extensions inside the Complex Numbers

theory *Field-Extension*

imports *Complex-Field-Extension HOL-Computational-Algebra.Field-as-Ring HOL-Computational-Algebra.Polynomial*
begin

Towards the transitivity of the Galois group of a rational polynomial on its complex roots, we develop the base isomorphism $\mathbb{Q}(\alpha) \cong \mathbb{Q}(\beta)$ for two roots α, β of the same irreducible polynomial. We work concretely: rational polynomials are *rat poly*, evaluated at a complex point through the embedding *complex-of-rat*. The evaluation map *evalQ a* is a ring homomorphism *rat poly* $\Rightarrow$ *complex*; its image is the subfield $\mathbb{Q}(a)$ and its kernel is the set of multiples of the minimal polynomial. Since *rat poly* is a principal ideal domain (a field-coefficient polynomial ring), the kernel characterization needs only Bézout's identity.

95.1 Evaluation of a rational polynomial at a complex point

definition *evalQ* :: *complex* $\Rightarrow$ *rat poly* $\Rightarrow$ *complex* **where**
evalQ a p = *poly (map-poly of-rat p) a*

lemma *map-poly-of-rat-add*: *map-poly of-rat (p + q) = map-poly of-rat p + map-poly of-rat q*
 $\langle \text{proof} \rangle$

lemma *map-poly-of-rat-mult*: *map-poly of-rat (p * q) = map-poly of-rat p * map-poly of-rat q*
 $\langle \text{proof} \rangle$

lemma *evalQ-add*: *evalQ a (p + q) = evalQ a p + evalQ a q*
 $\langle \text{proof} \rangle$

lemma *evalQ-mult*: *evalQ a (p * q) = evalQ a p * evalQ a q*
 $\langle \text{proof} \rangle$

lemma *evalQ-0*: *evalQ a 0 = 0* $\langle \text{proof} \rangle$

lemma *evalQ-1*: $\text{evalQ } a \ 1 = 1$ $\langle \text{proof} \rangle$

lemma *evalQ-diff*: $\text{evalQ } a \ (p - q) = \text{evalQ } a \ p - \text{evalQ } a \ q$
 $\langle \text{proof} \rangle$

The constant rational polynomial evaluates to the embedded constant; in particular *evalQ* fixes the rationals.

lemma *evalQ-const*: $\text{evalQ } a \ [:c:] = \text{of-rat } c$
 $\langle \text{proof} \rangle$

95.2 The kernel is generated by the minimal polynomial

If m is irreducible (over $\mathbb{Q}$) and both m and r vanish at a , then m divides r : by irreducibility m is coprime to any non-multiple, so Bezout would force $0 = 1$ under evaluation.

lemma *evalQ-kernel-dvd*:
assumes *m-irr*: *irreducible m* **and** *ma*: $\text{evalQ } a \ m = 0$ **and** *ra*: $\text{evalQ } a \ r = 0$
shows $m \text{ dvd } r$
 $\langle \text{proof} \rangle$

Consequently two roots of the same irreducible polynomial annihilate exactly the same rational polynomials.

lemma *evalQ-same-kernel*:
assumes *m-irr*: *irreducible m* **and** *ma*: $\text{evalQ } a \ m = 0$ **and** *mb*: $\text{evalQ } b \ m = 0$
and *ra*: $\text{evalQ } a \ r = 0$
shows $\text{evalQ } b \ r = 0$
 $\langle \text{proof} \rangle$

95.3 The simple extension $\mathbb{Q}(a)$ and the base isomorphism

The subfield $\mathbb{Q}(a)$ is the image of the evaluation map: every element is a rational polynomial expression in a .

definition *Qx* :: *complex* $\Rightarrow$ *complex set* **where**
 $\text{Qx } a = \text{range } (\text{evalQ } a)$

lemma *Qx-mem-evalQ* [*intro*]: $\text{evalQ } a \ p \in \text{Qx } a$
 $\langle \text{proof} \rangle$

lemma *Qx-self*: $a \in \text{Qx } a$
 $\langle \text{proof} \rangle$

lemma *Qx-Rats*: $\text{of-rat } c \in \text{Qx } a$
 $\langle \text{proof} \rangle$

For two roots a, b of the same irreducible polynomial m , the map sending $\text{evalQ } a \ p$ to $\text{evalQ } b \ p$ is well defined (independent of the representing polynomial p), by *evalQ-same-kernel*.

definition *simple-iso* :: *rat poly* $\Rightarrow$ *complex* $\Rightarrow$ *complex* $\Rightarrow$ *complex* $\Rightarrow$ *complex* **where**
simple-iso *m a b y* = *evalQ b (SOME p. evalQ a p = y)*

context

fixes *m* :: *rat poly* **and** *a b* :: *complex*

assumes *m-irr*: *irreducible m* **and** *ma*: *evalQ a m = 0* **and** *mb*: *evalQ b m = 0*

begin

Well-definedness: on the image, *simple-iso* agrees with *evalQ b*.

lemma *simple-iso-evalQ*: *simple-iso m a b (evalQ a p) = evalQ b p*
 $\langle \text{proof} \rangle$

lemma *simple-iso-add*:

$\llbracket x \in Qx\ a; y \in Qx\ a \rrbracket \implies \text{simple-iso}\ m\ a\ b\ (x + y) = \text{simple-iso}\ m\ a\ b\ x + \text{simple-iso}\ m\ a\ b\ y$
 $\langle \text{proof} \rangle$

lemma *simple-iso-mult*:

$\llbracket x \in Qx\ a; y \in Qx\ a \rrbracket \implies \text{simple-iso}\ m\ a\ b\ (x * y) = \text{simple-iso}\ m\ a\ b\ x * \text{simple-iso}\ m\ a\ b\ y$
 $\langle \text{proof} \rangle$

lemma *simple-iso-1*: *simple-iso m a b 1 = 1*
 $\langle \text{proof} \rangle$

lemma *simple-iso-0*: *simple-iso m a b 0 = 0*
 $\langle \text{proof} \rangle$

It fixes the rationals and sends *a* to *b*.

lemma *simple-iso-Rats*: *simple-iso m a b (of-rat c) = of-rat c*
 $\langle \text{proof} \rangle$

lemma *simple-iso-gen*: *simple-iso m a b a = b*
 $\langle \text{proof} \rangle$

It maps $\mathbb{Q}(a)$ onto $\mathbb{Q}(b)$.

lemma *simple-iso-image*: *simple-iso m a b ' Qx a = Qx b*
 $\langle \text{proof} \rangle$

end

end

96 The Galois correspondence between intermediate fields and subgroups

theory *Galois-Correspondence*

imports *Galois-Automorphism Order-Theory*

begin

Let $F \subseteq K$ be subfields of an ambient type-class field. There are two natural operations between the *intermediate fields* of the extension and the *subgroups* of its Galois group *field-auto* $K F$:

- *fixing* a set of automorphisms, $H \mapsto K^H$, the subfield of K on which every element of H acts as the identity;
- *the relative Galois group* of an intermediate field, $E \mapsto \text{field-auto } K E$, the automorphisms of K fixing E pointwise.

Both are *inclusion-reversing*: a bigger group of automorphisms fixes fewer points, and a bigger intermediate field admits fewer automorphisms fixing it. So this is an *antitone* adjoint pair, and the way to see it as a *Galois-Connection* — which is defined for *monotone* adjoints — is to order one of the two lattices by reverse inclusion. We order the subgroups by $\supseteq$; then both maps become isotone and the adjunction reads $K^H \subseteq E \iff H \supseteq \text{field-auto } K E$.

96.1 The fixed field of a set of automorphisms

The subfield of K fixed pointwise by every automorphism in H . We do not require H to be a subgroup.

The fixed field is a subfield by the generic closure theorem *field-auto-fixed-field-subfield*; the empty case is K itself.

theorem *fixed-field-subfield*:

assumes *Ksub*: Subfield K **and** H : $H \subseteq \text{field-auto } K F$

shows Subfield (fixed-field $K H$)

<proof>

The base field lies inside every fixed field: by definition each automorphism in the Galois group fixes F pointwise.

lemma *base-subset-fixed-field*:

assumes H : $H \subseteq \text{field-auto } K F$ **and** FK : $F \subseteq K$

shows $F \subseteq \text{fixed-field } K H$

<proof>

96.2 Intermediate fields and the relative Galois group

The intermediate fields of the extension $F \subseteq K$: the subfields lying between the two.

definition *inter-fields* :: 'a :: field set $\Rightarrow$ 'a set $\Rightarrow$ 'a set set

where *inter-fields* $K F \equiv \{E. \text{Subfield } E \wedge F \subseteq E \wedge E \subseteq K\}$

lemma *inter-fields-iff*:

$E \in \text{inter-fields } K F \iff \text{Subfield } E \wedge F \subseteq E \wedge E \subseteq K$

<proof>

The two ends of the extension are intermediate fields.

lemma *base-in-inter-fields:*

$\llbracket \text{Subfield } F; F \subseteq K \rrbracket \implies F \in \text{inter-fields } K$

<proof>

lemma *top-in-inter-fields:*

$\llbracket \text{Subfield } K; F \subseteq K \rrbracket \implies K \in \text{inter-fields } K$

<proof>

Taking the relative Galois group is inclusion-reversing: a bigger intermediate field imposes more conditions, so fewer automorphisms survive.

lemma *field-auto-antitone:* $E \subseteq E' \implies \text{field-auto } K \ E' \subseteq \text{field-auto } K \ E$

<proof>

The relative Galois group of an intermediate field is a subgroup of the whole Galois group: it consists of automorphisms of K , and fixing E pointwise implies fixing the smaller F pointwise.

theorem *field-auto-subgroup-of-Galois-group:*

assumes $K_{\text{sub}}: \text{Subfield } K$ **and** $FK: F \subseteq K$ **and** $E: E \in \text{inter-fields } K$

shows $\text{Subgroup } (\text{field-auto } K \ E) \ (\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K)$

<proof>

The set of subgroups of the Galois group, as the second of the two lattices.

definition *galois-subgroups* :: $'a :: \text{field set} \Rightarrow 'a \text{ set} \Rightarrow ('a \Rightarrow 'a) \text{ set set}$

where $\text{galois-subgroups } K \ F \equiv$

$\{H. \text{Subgroup } H \ (\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K)\}$

lemma *galois-subgroups-iff:*

$H \in \text{galois-subgroups } K \ F \longleftrightarrow \text{Subgroup } H \ (\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K)$

<proof>

lemma *galois-subgroups-subset:*

assumes $H \in \text{galois-subgroups } K \ F$ **shows** $H \subseteq \text{field-auto } K \ F$

<proof>

96.3 The adjunction

locale *galois-extension* =

fixes $K \ F :: 'a :: \text{field set}$

assumes $K_{\text{sub}}: \text{Subfield } K$ **and** $F_{\text{sub}}: \text{Subfield } F$ **and** $FK: F \subseteq K$

begin

Fixing lands in the intermediate fields: the fixed field of a subgroup is a subfield lying between F and K .

lemma *fixed-field-in-inter-fields:*

assumes $H: H \in \text{galois-subgroups } K F$
shows $\text{fixed-field } K H \in \text{inter-fields } K F$
 $\langle \text{proof} \rangle$

The adjunction. Both sides of the biconditional say the same thing — *every* automorphism in H fixes *every* point of E — so the proof is a matter of unfolding. This is the antitone (or *polarity*) form of an adjunction: both maps reverse inclusion, and each side bounds one argument by the image of the other.

It is worth being precise about which biconditional is formal, because the other one is a deep theorem. The statement proved here bounds E *above* by a fixed field and H *above* by a Galois group. The reverse reading — $\text{fixed-field } K H \subseteq E \longleftrightarrow \text{field-auto } K E \subseteq H$ — is *not* a formal consequence: its left-to-right direction asserts that every automorphism fixing E already lies in H , which is Artin’s theorem and needs H finite.

theorem *fixed-field-galois:*

assumes $E: E \in \text{inter-fields } K F$ **and** $H: H \in \text{galois-subgroups } K F$
shows $E \subseteq \text{fixed-field } K H \longleftrightarrow H \subseteq \text{field-auto } K E$
 $\langle \text{proof} \rangle$

The two *unit* inequalities, which hold unconditionally and in both directions. These are the parts of the correspondence that need no hypotheses; it is their *converses* that require normality and separability.

lemma *le-fixed-field-galois-group:*

assumes $E: E \in \text{inter-fields } K F$
shows $E \subseteq \text{fixed-field } K (\text{field-auto } K E)$
 $\langle \text{proof} \rangle$

lemma *le-galois-group-fixed-field:*

assumes $H: H \in \text{galois-subgroups } K F$
shows $H \subseteq \text{field-auto } K (\text{fixed-field } K H)$
 $\langle \text{proof} \rangle$

96.4 The Galois connection

Both maps reverse inclusion, so to view the pair as a *Galois-Connection* — whose adjoints are *isotone* — we order the subgroup lattice by *reverse* inclusion. Then the adjunction *fixed-field-galois* is literally the *galois* axiom, with *field-auto* K as the lower adjoint out of the intermediate fields and *fixed-field* K as the upper adjoint back.

Note the direction: *field-auto* K is the *lower* adjoint here, even though it is the map *into* the groups, because the group side carries the reversed order. Inclusion is a partial order on any set of sets, and so is its converse.

Inclusion and reverse inclusion are partial orders on the two lattices.

lemma *partial-order-inter-fields: Partial-Order (inter-fields $K F$) ($\subseteq$)*
 $\langle \text{proof} \rangle$

lemma *partial-order-galois-subgroups*: *Partial-Order* (*galois-subgroups* $K F$) ($\lambda H H'$.
 $H' \subseteq H$)
 $\langle \text{proof} \rangle$

theorem *galois-correspondence*:

Galois-Connection

(*inter-fields* $K F$) ($\subseteq$)

(*galois-subgroups* $K F$) ($\lambda H H'. H' \subseteq H$)

(*field-auto* K) (*fixed-field* K)

$\langle \text{proof} \rangle$

The consequences, specialised to the correspondence. As in *Ideal-Extension* these come from a local **interpretation** of the theorem rather than a **sublocale** declaration.

context

begin

interpretation *GC*: *Galois-Connection*

inter-fields $K F$ ($\subseteq$) *galois-subgroups* $K F$ $\lambda H H'. H' \subseteq H$

field-auto K *fixed-field* K

$\langle \text{proof} \rangle$

Both maps are inclusion-reversing on their lattices — the isotonicity of the connection, read back through the reversed order on subgroups.

corollary *field-auto-mono*:

$\llbracket E \in \text{inter-fields } K F; E' \in \text{inter-fields } K F; E \subseteq E' \rrbracket$

$\implies \text{field-auto } K E' \subseteq \text{field-auto } K E$

$\langle \text{proof} \rangle$

corollary *fixed-field-mono*:

$\llbracket H \in \text{galois-subgroups } K F; H' \in \text{galois-subgroups } K F; H' \subseteq H \rrbracket$

$\implies \text{fixed-field } K H \subseteq \text{fixed-field } K H'$

$\langle \text{proof} \rangle$

The *semi-inverse* laws: one round trip and back is the identity. These are the strongest fixed-point statements available without finiteness — they say the closure and kernel operators are idempotent, *not* that either is the identity.

corollary *field-auto-fixed-field-field-auto*:

$E \in \text{inter-fields } K F \implies \text{field-auto } K (\text{fixed-field } K (\text{field-auto } K E)) = \text{field-auto } K E$

$\langle \text{proof} \rangle$

corollary *fixed-field-field-auto-fixed-field*:

$H \in \text{galois-subgroups } K F \implies \text{fixed-field } K (\text{field-auto } K (\text{fixed-field } K H)) = \text{fixed-field } K H$

$\langle \text{proof} \rangle$

end

end

Scope and downstream completion. This theory deliberately stops at the representation-independent, inclusion-reversing adjunction, packaged as a *Galois-Connection*. No separability, normality or finiteness hypothesis belongs in this layer: the unit, counit and semi-inverse laws are the reusable order-theoretic API.

end

97 Normal and separable complex field extensions

theory *Galois-Normality*

imports *Iso-Extension-Tower Extension-Properties*

begin

The generic extension predicates and their root-set API live in *Extension-Properties*. This theory supplies the complex-specific characteristic-zero and algebraically-closed-field arguments needed by the Galois development: normality is witnessed by root transfer, while separability follows from the nonvanishing characteristic. The root-transfer theorem below is the useful bridge: an isomorphism between two simple subextensions extends across the finite root set, and therefore becomes an automorphism of the complex splitting field.

In characteristic zero every algebraic element is separable. This is the only separability input needed here; keeping the Bezout argument local avoids importing the Galois-degree and symmetric-generation developments into the normality layer.

lemma (in *Subfield*) *galois-normality-poly-over-pderiv*:

assumes $p \in \text{poly-over } K$

shows $\text{pderiv } p \in \text{poly-over } K$

<proof>

lemma *rsquarefree-minpoly-char-0*:

fixes $F :: 'a :: \text{field-char-0 set}$ **and** $a :: 'a$

assumes $\text{sfF}: \text{Subfield } F$ **and** $\text{alg}: \text{algebraic-over } F a$

shows $\text{rsquarefree } (\text{minpoly } F a)$

<proof>

In characteristic zero, the distinct complex roots of an algebraic element's minimal polynomial are counted by its simple extension degree. This elementary root-count bridge is kept with the separability API so later Galois results need not import the Galois-action machinery.

lemma *card-minpoly-roots-eq-ext-degree*:

fixes $F :: \text{complex set}$ **and** $a :: \text{complex}$

assumes $\text{sfF}: \text{Subfield } F$ **and** $\text{alg}: \text{algebraic-over } F a$

shows $\text{card } \{r. \text{poly } (\text{minpoly } F \ a) \ r = 0\} = \text{ext-degree } F \ a$
 $\langle \text{proof} \rangle$

Every complex extension is separable whenever the element under consideration is algebraic.

theorem *complex-extension-separable:*

assumes sfF : *complex-subfield* F

shows *separable-extension* $K \ F$

$\langle \text{proof} \rangle$

The root-transfer argument needs only the elementary homomorphism facts from the transitivity development. They are kept here as a small local interface so that this normality layer does not depend on the later Galois-action theory.

lemma *normality-field-hom-on-id:*

assumes *Subfield* K

shows *field-hom-on* $K \ (\lambda x. x)$

$\langle \text{proof} \rangle$

lemma *normality-map-poly-fix:*

fixes $f :: 'a :: \text{field} \Rightarrow 'a$

assumes hom : *field-hom-on* $K \ f$ **and** sfF : *Subfield* F

and p : $p \in \text{poly-over } F$ **and** fixF : $\bigwedge x. x \in F \Longrightarrow f \ x = x$

shows *map-poly* $f \ p = p$

$\langle \text{proof} \rangle$

lemma *normality-hom-preserves-roots:*

fixes $f :: 'a :: \text{field} \Rightarrow 'a$

assumes hom : *field-hom-on* $K \ f$ **and** sfF : *Subfield* F **and** FK : $F \subseteq K$

and fixF : $\bigwedge x. x \in F \Longrightarrow f \ x = x$

and Phi : $\text{Phi} \in \text{poly-over } F$ **and** r : $r \in K$ **and** root : *poly* $\text{Phi} \ r = 0$

shows *poly* $\text{Phi} \ (f \ r) = 0$

$\langle \text{proof} \rangle$

Extend a conjugacy between two simple subextensions across the root set of a splitting field. The resulting map permutes that finite root set, hence maps the generated field onto itself.

theorem *splitting-field-root-transfer:*

fixes $F \ K :: \text{complex set}$ **and** $p \ m :: \text{complex poly}$ **and** $a \ b :: \text{complex}$

assumes split : *splitting-field* $F \ p \ K$

and sfF : *Subfield* F

and alg : *algebraic-over* $F \ a$

and aK : $a \in K$

and minp : *is-minpoly* $F \ a \ m$

and root : *poly* $m \ b = 0$

shows $\exists \sigma \in \text{field-auto } K \ F. \ \sigma \ a = b$

$\langle \text{proof} \rangle$

A splitting field is normal in the root-theoretic sense. The proof does not assume that the polynomial used to construct the splitting field is irreducible: the finite

root set is only the carrier across which the simple conjugacy is extended.

theorem *splitting-field-normal*:
assumes *split*: *splitting-field* F p K **and** *sfF*: *complex-subfield* F
shows *normal-extension* K F
 $\langle proof \rangle$

end

98 Fixed fields of finite normal and separable extensions

theory *Galois-Fixed-Field*
imports *Galois-Correspondence* *Galois-Normality*
begin

A splitting field remains a splitting field after enlarging its base to an intermediate field. This is the representation bridge needed below: the root set is unchanged, while the base part of the generated subfield grows from F to E .

lemma *splitting-field-over-intermediate*:
fixes F E K :: *complex set* **and** p :: *complex poly*
assumes *split*: *splitting-field* F p K
and E : $E \in \text{inter-fields } K$ F
shows *splitting-field* E p K
 $\langle proof \rangle$

lemma *card-gt-one-imp-exists-mem-ne*:
fixes A :: *'a set* **and** x :: *'a* **and** n :: *nat*
assumes xA : $x \in A$
and $cardA$: $\text{card } A = n$
and ngt : $1 < n$
shows $\exists b. b \in A \wedge b \neq x$
 $\langle proof \rangle$

The closure half of the correspondence is most transparent as a separation statement. If an element of a finite splitting field is outside the intermediate field, its minimal polynomial has a second root: characteristic zero gives distinct roots, and degree one is exactly the criterion for already belonging to the base. The root-transfer theorem then supplies an automorphism fixing the intermediate field that moves the element.

The explicit *algebraic-extension* assumption records the finite-extension input without importing Artin or an algebraic-closure construction. A later finite-generation layer can discharge it for the intended classes of root-generated extensions.

theorem *fixed-field-eq-of-splitting-field*:
fixes K F E :: *complex set* **and** p :: *complex poly*
assumes *split*: *splitting-field* F p K

```

    and E: E ∈ inter-fields K F
    and alg: algebraic-extension K E
    shows fixed-field K (field-auto K E) = E
  ⟨proof⟩

end

```

99 Primitive elements of finite field extensions

```

theory Primitive-Element
  imports Algebraic-Transitivity Extension-Properties Field-Mult-Cyclic Galois-Transitivity
begin

```

A primitive element is an element whose simple extension is the whole target field. The definition is deliberately just the equality with *eval-img*: the hypotheses that make this set a field belong to the surrounding extension theorem, rather than being hidden in the predicate itself.

```

definition primitive-element :: 'a :: field set ⇒ 'a set ⇒ 'a ⇒ bool
  where primitive-element K L a ≡ L = eval-img K a

```

```

lemma primitive-elementI: L = eval-img K a ⇒ primitive-element K L a
  ⟨proof⟩

```

```

lemma primitive-elementD: primitive-element K L a ⇒ L = eval-img K a
  ⟨proof⟩

```

```

lemma
  assumes primitive-element K L a Subfield K
  shows primitive-element-mem: a ∈ L and primitive-element-base: K ⊆ L
  ⟨proof⟩

```

99.1 The two-generator theorem over an infinite base

The classical separating-linear-combination argument is the substantive core of the primitive element theorem. For separable algebraic elements a and b , only finitely many scalars c can make two distinct pairs of conjugates have the same value under $(x, y) \mapsto x + c*y$. An infinite base field therefore supplies a scalar outside that exceptional set. The resulting element $a + c*b$ recovers both generators.

```

theorem primitive-element-two-generators:
  fixes F :: 'a :: alg-closed-field set
  assumes sfF: Subfield F and infF: infinite F
    and alg-a: algebraic-over F a and alg-b: algebraic-over F b
    and sep-a: rsquarefree (minpoly F a) and sep-b: rsquarefree (minpoly F b)
  shows ∃ theta. eval-img (eval-img F a) b = eval-img F theta
  ⟨proof⟩

```

99.2 Finite separable extensions are simple

Over an infinite base, a finite algebraic generating set can be collapsed one generator at a time. Separability is asked of the containing extension, so it applies both to the next generator and to the primitive element already obtained for the preceding generators.

lemma *finite-separable-generators-primitive:*
fixes $K L A :: 'a :: \text{alg-closed-field set}$
assumes $T: \text{finite-subfield-tower } K L$
and $\text{inf}K: \text{infinite } K$ **and** $\text{fin}A: \text{finite } A$ **and** $AL: A \subseteq L$
and $\text{sep}: \text{separable-extension } L K$
shows $\exists \text{theta} \in L. \text{generate-field } (K \cup A) = \text{eval-img } K \text{ theta}$
<proof>

Finite fields have a particularly clean primitive-element proof. The multiplicative group of the target field is cyclic, so a generator of its nonzero elements already generates the field as a simple extension over every subfield. This is the finite-field instance of the primitive element theorem and is the useful bridge for the concrete finite-field theories.

theorem *finite-field-extension-is-simple:*
assumes $KL: \text{subfield-tower } K L$ **and** $\text{fin}L: \text{finite } L$
shows $\exists a \in L. \text{primitive-element } K L a$
<proof>

The full primitive-element theorem now follows from a basis. If the base is finite, the finite coordinate space makes the target field finite and the multiplicative-cyclic proof applies. If the base is infinite, the preceding finite-generator theorem collapses the basis to one element.

theorem *finite-separable-extension-is-simple:*
fixes $K L :: 'a :: \text{alg-closed-field set}$
assumes $T: \text{finite-subfield-tower } K L$ **and** $\text{sep}: \text{separable-extension } L K$
shows $\exists a \in L. \text{primitive-element } K L a$
<proof>

lemma *(in finite-subfield-tower) primitive-element-degree:*
assumes $\text{primitive-element } K L a$
shows $\text{extension-degree} = \text{ext-degree } K a$
<proof>

end

100 The order of the Galois group of a finite normal separable extension

theory *Galois-Degree*
imports *Galois-Action Extension-Properties Primitive-Element*
begin

For a *simple normal* extension $K = F(a)$ of complex fields (all conjugates of a — the roots of its minimal polynomial — already lying in K), the order of the Galois group equals the number of those conjugates, and, in characteristic 0 where the minimal polynomial is separable, this equals *ext-degree* $F a$. The argument is the classical bijection $\lambda\sigma. \sigma a$ between *field-auto* $K F$ and the conjugates of a : it is injective (an F -automorphism of $F a$ is determined by where it sends a), lands among the conjugates (*field-auto-maps-root*), and is onto (the Galois action is transitive on the roots of an irreducible polynomial). The simple-extension calculation is the reusable core. The final theorem combines it with the primitive-element theorem and the basis definition of extension degree to obtain $|Gal(K/F)| = [K:F]$ for every finite normal separable extension in an algebraically closed ambient field.

100.1 The minimal polynomial is irreducible over the base

A minimal polynomial is irreducible over F : it is nonzero of positive degree and, by minimality, admits no factorisation over F into two positive-degree factors.

lemma *irreducible-over-minpoly*:

assumes *sfF*: Subfield F **and** *alg*: algebraic-over $F a$

shows *irreducible-over* F (*minpoly* $F a$)

<proof>

100.2 An F -automorphism of $F(a)$ is determined by its value at a

Two F -automorphisms of the simple extension *eval-img* $F a$ that agree at a agree everywhere. Every element of *eval-img* $F a$ is *poly* $p a$ for some $p \in$ *poly-over* F , and an F -automorphism commutes with such evaluation (*field-hom-on.poly-map-poly-hom*, together with *map-poly-fix* since the coefficients lie in F); so the common value at a fixes the value everywhere.

lemma *field-auto-determined-by-gen*:

assumes *sfF*: Subfield F **and** *alg*: algebraic-over $F a$

and *s*: $\sigma \in$ *field-auto* (*eval-img* $F a$) F **and** *t*: $\tau \in$ *field-auto* (*eval-img* $F a$) F

and *eq*: $\sigma a = \tau a$

shows $\sigma = \tau$

<proof>

100.3 The order of the Galois group of a simple normal extension

Normality of the simple extension: all conjugates of a lie in *eval-img* $F a$. Under this hypothesis, $\lambda\sigma. \sigma a$ is a bijection from *field-auto* (*eval-img* $F a$) F to the conjugate set, so the group order equals the number of conjugates.

theorem *galois-simple-normal-card*:

fixes $F ::$ complex set **and** $a ::$ complex

assumes *sfF*: Subfield *F* **and** *alg*: algebraic-over *F a*
and *normal*: $\{r. \text{poly} (\text{minpoly } F a) r = 0\} \subseteq \text{eval-img } F a$
shows $\text{card} (\text{field-auto} (\text{eval-img } F a) F) = \text{card} \{r. \text{poly} (\text{minpoly } F a) r = 0\}$
 $\langle \text{proof} \rangle$

100.4 Separability in characteristic 0: the number of conjugates is the degree

poly-over is closed under formal differentiation: the coefficients of *pderiv p* are integer multiples of coefficients of *p* (*coeff-pderiv*), and a subfield contains all *of-nat n*.

lemma (in Subfield) *poly-over-pderiv*:
assumes *p* ∈ *poly-over K*
shows *pderiv p* ∈ *poly-over K*
 $\langle \text{proof} \rangle$

In characteristic 0 the minimal polynomial is *separable* (squarefree): it shares no root with its derivative. As *pderiv m* has smaller degree than the irreducible *m*, it is not a multiple of *m*, so Bézout over *F* gives $u * m + v * \text{pderiv } m = 1$; evaluating this *polynomial identity* at any root *r* of *m* yields $\text{poly } v r * \text{poly} (\text{pderiv } m) r = 1$, so $\text{poly} (\text{pderiv } m) r \neq 0$.

lemma *rsquarefree-minpoly*:
fixes *F* :: 'a :: field-char-0 set **and** *a* :: 'a
assumes *sfF*: Subfield *F* **and** *alg*: algebraic-over *F a*
shows *rsquarefree* (*minpoly F a*)
 $\langle \text{proof} \rangle$

Hence, over *complex*, a separable minimal polynomial that splits has exactly *degree* (*minpoly F a*) distinct roots: it decomposes as a product of the distinct linear factors $[-z, 1:]$ (*complex-poly-decompose-rsquarefree*), and the degree of that product is the number of factors.

lemma *card-conjugates-eq-degree*:
fixes *F* :: complex set **and** *a* :: complex
assumes *sfF*: Subfield *F* **and** *alg*: algebraic-over *F a*
shows $\text{card} \{r. \text{poly} (\text{minpoly } F a) r = 0\} = \text{ext-degree } F a$
 $\langle \text{proof} \rangle$

100.5 The headline: order of the Galois group of a simple normal extension

The order of the Galois group of a simple normal extension equals the extension degree. Combining the bijection *galois-simple-normal-card* with separability *card-conjugates-eq-degree*: for a simple normal complex extension *F a*,

$$|\text{field_auto } (F(a)) F| = \text{ext_degree } F a = \text{deg}(\text{minpoly } F a).$$

This is $|\text{Gal}(K/F)| = [K:F]$ in the simple normal (separable) case.

theorem *galois-simple-normal-degree*:
fixes $F :: \text{complex set}$ **and** $a :: \text{complex}$
assumes $\text{sfF: Subfield } F$ **and** $\text{alg: algebraic-over } F \ a$
and $\text{normal: } \{r. \text{poly } (\text{minpoly } F \ a) \ r = 0\} \subseteq \text{eval-img } F \ a$
shows $\text{card } (\text{field-auto } (\text{eval-img } F \ a) \ F) = \text{ext-degree } F \ a$
 $\langle \text{proof} \rangle$

100.6 Generic finite normal and separable extensions

Normality of a simple extension identifies its automorphisms with the roots of the minimal polynomial. The proof is the generic root-action argument: an automorphism is determined by the generator, maps the generator to a root, and transitivity supplies an automorphism for each root.

theorem *galois-simple-normal-card-alg-closed*:
fixes $F :: 'a :: \text{alg-closed-field set}$ **and** $a :: 'a$
assumes $\text{sfF: Subfield } F$ **and** $\text{alg: algebraic-over } F \ a$
and $\text{normal: } \{r. \text{poly } (\text{minpoly } F \ a) \ r = 0\} \subseteq \text{eval-img } F \ a$
shows $\text{card } (\text{field-auto } (\text{eval-img } F \ a) \ F) = \text{card } \{r. \text{poly } (\text{minpoly } F \ a) \ r = 0\}$
 $\langle \text{proof} \rangle$

The simple normal argument is now available over every algebraically closed field. The normality hypothesis says that all roots of the minimal polynomial lie in the simple extension; separability supplies squarefreeness, and the generic root action supplies the matching automorphism count.

theorem *galois-simple-normal-separable-degree*:
fixes $F :: 'a :: \text{alg-closed-field set}$ **and** $a :: 'a$
assumes $\text{sfF: Subfield } F$ **and** $\text{alg: algebraic-over } F \ a$
and $\text{normal: } \{r. \text{poly } (\text{minpoly } F \ a) \ r = 0\} \subseteq \text{eval-img } F \ a$
and $\text{sep: separable-extension } (\text{eval-img } F \ a) \ F$
shows $\text{card } (\text{field-auto } (\text{eval-img } F \ a) \ F) = \text{ext-degree } F \ a$
 $\langle \text{proof} \rangle$

A finite normal separable extension is simple by the primitive-element theorem. Normality then puts every conjugate of the primitive element back in the target field, so the preceding simple theorem computes the whole automorphism group. The basis theorem identifies the simple degree with the intrinsic extension degree.

context
fixes $F \ K :: 'a :: \text{alg-closed-field set}$
assumes $\text{finite: finite-subfield-tower } F \ K$
begin

interpretation $T: \text{finite-subfield-tower } F \ K \ \langle \text{proof} \rangle$

theorem *finite-normal-separable-field-auto*:
assumes $\text{normal: normal-extension } K \ F$ **and** $\text{sep: separable-extension } K \ F$
shows $\text{finite } (\text{field-auto } K \ F)$
 $\langle \text{proof} \rangle$

```

theorem finite-normal-separable-galois-degree:
  assumes normal: normal-extension  $K\ F$  and sep: separable-extension  $K\ F$ 
  shows  $\text{card } (\text{field-auto } K\ F) = T.\text{extension-degree}$ 
  <proof>

end

end

```

101 Field-side inverse and degree formulas for simple normal extensions

```

theory Galois-Simple-Degree
imports Galois-Fixed-Field Algebraic-Transitivity Galois-Degree
begin

```

101.1 Algebraicity of generated splitting fields

The generic algebraicity closure for generated fields lives in *Algebraic-Transitivity*. This complex-specific wrapper applies it to the root set of a splitting-field presentation.

Complex splitting fields are algebraic over their base. Their generators are the root set of the defining nonzero polynomial, and the preceding closure result applies to that set directly (in fact, it does not need the root set's finiteness).

```

lemma splitting-field-algebraic-extension:
  fixes  $F\ K :: \text{complex set}$  and  $q :: \text{complex poly}$ 
  assumes split: splitting-field  $F\ q\ K$  and sfF: complex-subfield  $F$ 
  shows algebraic-extension  $K\ F$ 
  <proof>

```

```

theorem splitting-field-normal-separable:
  fixes  $F\ K :: \text{complex set}$  and  $q :: \text{complex poly}$ 
  assumes split: splitting-field  $F\ q\ K$  and sfF: complex-subfield  $F$ 
  shows normal-extension  $K\ F \wedge$  separable-extension  $K\ F$ 
  <proof>

```

With the finite-generation bridge in place, the closure theorem no longer needs an explicit algebraicity premise for an intermediate field of a complex splitting field.

```

theorem fixed-field-eq-of-complex-splitting-field:
  fixes  $F\ E\ K :: \text{complex set}$  and  $q :: \text{complex poly}$ 
  assumes split: splitting-field  $F\ q\ K$ 
  and sfF: complex-subfield  $F$ 
  and  $E$ :  $E \in \text{inter-fields } K\ F$ 
  shows fixed-field  $K$  ( $\text{field-auto } K\ E$ ) =  $E$ 

```

<proof>

The simple normal case is the degree-bearing part of the field-side correspondence. The existing Galois-degree theorem counts the automorphisms; this theory connects its hypotheses to the splitting-field presentation used by the fixed-field closure theorem.

lemma *splitting-field-finite-galois-group:*

fixes $F\ K :: \text{complex set}$ **and** $q :: \text{complex poly}$
assumes *split*: *splitting-field* $F\ q\ K$ **and** *sfF*: *complex-subfield* F
shows *finite* (*field-auto* $K\ F$)

<proof>

lemma *simple-normal-splitting-field:*

fixes $F :: \text{complex set}$ **and** $a :: \text{complex}$
assumes *sfF*: *Subfield* F **and** *alg*: *algebraic-over* $F\ a$
and *normal*: $\{r. \text{poly } (\text{minpoly } F\ a)\ r = 0\} \subseteq \text{eval-img } F\ a$
shows *splitting-field* $F\ (\text{minpoly } F\ a)\ (\text{eval-img } F\ a)$

<proof>

For a simple normal complex extension, the field-side inverse law and the exact order formula can be consumed together. The subgroup-side inverse law is deliberately not reproved here; it is the separate Artin theorem.

theorem *simple-normal-field-inverse-and-degree:*

fixes $F :: \text{complex set}$ **and** $a :: \text{complex}$
assumes *sfF*: *complex-subfield* F **and** *alg*: *algebraic-over* $F\ a$
and *normal*: $\{r. \text{poly } (\text{minpoly } F\ a)\ r = 0\} \subseteq \text{eval-img } F\ a$
shows *fixed-field* (*eval-img* $F\ a$) (*field-auto* (*eval-img* $F\ a$) F) = $F \wedge$
card (*field-auto* (*eval-img* $F\ a$) F) = *ext-degree* $F\ a$

<proof>

end

102 Generic restriction of field automorphisms

theory *Galois-Restriction*

imports *Galois-Automorphism Solvable-Transfer Galois-Action*

begin

For a base subset $F \subseteq E \subseteq K$, where E and K are subfields, stability of the intermediate carrier under the base-field automorphisms makes restriction a homomorphism between the two corresponding automorphism groups. This theory contains the reusable carrier-set bookkeeping; splitting-field and polynomial-extension arguments remain in the theories that consume it.

definition *galois-restriction* ::

$'a :: \text{field set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow$
 $(('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a))$

where *galois-restriction* ($K :: 'a \text{ set}$) $E\ F = \text{restrict } (\lambda \sigma. \text{restrict } \sigma\ E)\ (\text{field-auto } K\ F)$

lemma *galois-restriction-apply* [simp]:
 $\sigma \in \text{field-auto } K \ F \implies$
 $\text{galois-restriction } K \ E \ F \ \sigma = \text{restrict } \sigma \ E$
 ⟨proof⟩

context
fixes $K \ E \ F :: 'a :: \text{field set}$
assumes $K: \text{Subfield } K$ **and** $\text{sfE}: \text{Subfield } E$
and $\text{FE}: F \subseteq E$ **and** $\text{EK}: E \subseteq K$
and *stable*: $\bigwedge \sigma. \sigma \in \text{field-auto } K \ F \implies \sigma \restriction E = E$
begin

lemma *field-auto-bij*: $\sigma \in \text{field-auto } K \ F \implies \text{bij-betw } \sigma \ E \ E$
 ⟨proof⟩

Restriction inherits the field laws and fixed-base condition, while stability supplies the required bijection of the intermediate carrier.

lemma *gal-res-mem*:
assumes $s: \sigma \in \text{field-auto } K \ F$
shows $\text{restrict } \sigma \ E \in \text{field-auto } E \ F$
 ⟨proof⟩

Restriction turns composition in K into composition in E .

lemma *gal-res-compose*:
assumes $s: \sigma \in \text{field-auto } K \ F$ **and** $t: \tau \in \text{field-auto } K \ F$
shows $\text{restrict } (\text{compose } K \ \sigma \ \tau) \ E = \text{compose } E \ (\text{restrict } \sigma \ E) \ (\text{restrict } \tau \ E)$
 ⟨proof⟩

Restriction sends the identity of K to the identity of E .

lemma *gal-res-unit*: $\text{restrict } (\text{identity } K) \ E = \text{identity } E$
 ⟨proof⟩

theorem *galois-restriction-hom*:
 $\text{group-homomorphism } (\text{restrict } (\lambda \sigma. \text{restrict } \sigma \ E)) \ (\text{field-auto } K \ F))$
 $(\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K)$
 $(\text{field-auto } E \ F) \ (\text{compose } E) \ (\text{identity } E)$
 ⟨proof⟩

theorem *galois-restriction-epimorphism*:
assumes *onto*: $\text{galois-restriction } K \ E \ F \restriction \text{field-auto } K \ F = \text{field-auto } E \ F$
shows $\text{group-epimorphism } (\text{galois-restriction } K \ E \ F)$
 $(\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K)$
 $(\text{field-auto } E \ F) \ (\text{compose } E) \ (\text{identity } E)$
 ⟨proof⟩

theorem *galois-restriction-Ker*:
 $\text{group-homomorphism.Ker } (\text{restrict } (\lambda \sigma. \text{restrict } \sigma \ E)) \ (\text{field-auto } K \ F))$
 $(\text{field-auto } K \ F) \ (\text{identity } E) = \text{field-auto } K \ E$

<proof>

The standard first-isomorphism theorem applies to this generic restriction map.

interpretation *restriction-core: group-homomorphism*

galois-restriction K E F
field-auto K F compose K identity K
field-auto E F compose E identity E

<proof>

theorem *galois-restriction-factor-isomorphism:*

(restriction-core.kernel.Factor-Group,
restriction-core.kernel.quotient-composition,
restriction-core.kernel.Class (identity K)) $\cong_G$
(galois-restriction K E F ‘ field-auto K F, compose E, identity E)
<proof>

102.1 The inductive step: solvability climbs the tower

With E stable, solvability of the relative kernel and the intermediate-field automorphism group implies solvability of the full automorphism group. The restriction map is corestricted to its image; the group quotient is handled by the standard first-isomorphism machinery.

theorem *galois-solvable-tower-step:*

assumes *solvKE: Group.solvable (field-auto K E) (compose K) (identity K)*
and *solvEF: Group.solvable (field-auto E F) (compose E) (identity E)*
shows *Group.solvable (field-auto K F) (compose K) (identity K)*
<proof>

end

102.2 Complex splitting-field extension of generic restriction

Surjectivity of restriction for a generated root field. Let $F \subseteq E \subseteq K$ with K generated by E and the roots of an F -polynomial. The finite homomorphism-extension theorem supplies an extension of every F -automorphism of E .

theorem *galois-restriction-surjective:*

fixes *K E F :: 'a :: field set and Phi :: 'a poly*
assumes *sfE: Subfield E and sfF: Subfield F and FE: F $\subseteq$ E*
and *Phi: Phi $\in$ poly-over F*
and *Kdef: K = generate-field (E $\cup$ {r. poly Phi r = 0})*
and *finR: finite {r. poly Phi r = 0}*
and *algR: $\bigwedge r. \text{poly } \Phi r = 0 \implies \text{algebraic-over } E r$*
and *closed: $\bigwedge q :: 'a \text{ poly. } 0 < \text{degree } q \implies \exists b. \text{poly } q b = 0$*
and *t: $\tau \in \text{field-auto } E F$*
shows *$\exists \sigma \in \text{field-auto } K F. \forall x \in E. \sigma x = \tau x$*
<proof>

102.3 A normal solvable tower has a solvable Galois group

Given a fixed complex splitting field K and a chain of intermediate fields $F = F_0 \subseteq F_1 \subseteq \dots \subseteq F_n = K$ in which each F_i is *stable* under *field-auto* K F and each step's relative Galois group *field-auto* F_{i+1} F_i is solvable, the whole Galois group *field-auto* K F is solvable. This chains the inductive step *galois-solvable-tower-step* down the tower, the base case being the trivial group *field-auto* K K .

102.3.1 The trivial Galois group

An automorphism of K fixing all of K is the identity, so *field-auto* K K is trivial.

lemma *field-auto-self-trivial*:

fixes $K :: \text{complex set}$

assumes K : *Subfield* K

shows *field-auto* K $K = \{\text{identity } K\}$

<proof>

lemma *solvable-field-auto-self*:

fixes $K :: \text{complex set}$

assumes K : *Subfield* K

shows *Group.solvable* (*field-auto* K K) (*compose* K) (*identity* K)

<proof>

102.3.2 Normal solvable towers

solvable-tower K F F s: the chain starting at F and following F s, with fixed top K , in which every field is a subfield of K , every intermediate field is stable under *field-auto* K F , and every step has a solvable relative Galois group. (The empty tower requires $F = K$.)

fun *solvable-tower* :: *complex set* $\Rightarrow$ *complex set* $\Rightarrow$ *complex set list* $\Rightarrow$ *bool* **where**

solvable-tower K F [] $\longleftrightarrow F = K$

| *solvable-tower* K F ($G \# G$ s) $\longleftrightarrow$

Subfield $F \wedge \text{Subfield } G \wedge F \subseteq G \wedge G \subseteq K \wedge$

$(\forall \sigma \in \text{field-auto } K \text{ } F. \sigma \text{ ' } G = G) \wedge$

Group.solvable (*field-auto* G F) (*compose* G) (*identity* G) $\wedge$

solvable-tower K G G s

102.3.3 The assembly

A normal solvable tower has a solvable Galois group. By induction down the tower: the top K has the trivial (hence solvable) group *field-auto* K K ; each step combines the solvable relative group *field-auto* G F with the solvable *field-auto* K G (from the induction hypothesis) via *galois-solvable-tower-step*, using stability of G under *field-auto* K F .

theorem *solvable-tower-imp-solvable*:

assumes K : *Subfield* K


```

shows solvable-tower  $K F Fs \implies \text{Subfield } F$ 
       $\implies \text{Group.solvable } (\text{field-auto } K F) (\text{compose } K) (\text{identity } K)$ 
<proof>

end

```

103 Restriction along a stable intermediate field of a splitting field

```

theory Galois-Restriction-Splitting
imports Galois-Simple-Degree Galois-Restriction
begin

```

The general restriction theory exposes the homomorphism and its kernel, but leaves the surjectivity argument in terms of a generated root field. This theory supplies the missing representation bridge for the public *splitting-field* interface. The stability hypothesis is essential: without it, restricting an F -automorphism of K need not map the intermediate field E to itself, so the restriction does not land in *field-auto* $E F$.

The roots of a splitting-field polynomial give exactly the generated-field presentation required by the finite homomorphism-extension theorem.

```

lemma splitting-field-generate-field-over:
  fixes  $K E :: \text{complex set}$  and  $p :: \text{complex poly}$ 
  assumes split: splitting-field  $E p K$ 
  shows  $K = \text{generate-field } (E \cup \{r. \text{poly } p r = 0\})$ 
<proof>

```

For a splitting field over E , restriction onto E is surjective onto the F -automorphisms of E , provided the defining polynomial is already over F . This is the form used by the radical-solvability descent: the larger group fixes F , while the splitting-field generators are algebraic over E .

```

theorem galois-restriction-image-of-splitting-field:
  fixes  $K E F :: \text{complex set}$  and  $p :: \text{complex poly}$ 
  assumes split: splitting-field  $E p K$ 
  and sfE: Subfield  $E$  and sfF: Subfield  $F$  and FE:  $F \subseteq E$ 
  and pF:  $p \in \text{poly-over } F$ 
  and stable:  $\bigwedge \sigma. \sigma \in \text{field-auto } K F \implies \sigma \restriction E = \text{id}_E$ 
  shows galois-restriction  $K E F \restriction = \text{field-auto } K F \restriction = \text{field-auto } E F$ 
<proof>

```

Package the preceding image equality as an epimorphism.

```

theorem galois-restriction-epimorphism-of-splitting-field:
  fixes  $K E F :: \text{complex set}$  and  $p :: \text{complex poly}$ 
  assumes split: splitting-field  $E p K$ 
  and sfE: Subfield  $E$  and sfF: Subfield  $F$  and FE:  $F \subseteq E$ 
  and pF:  $p \in \text{poly-over } F$ 

```

and stable: $\bigwedge \sigma. \sigma \in \text{field-auto } K \ F \implies \sigma \restriction E = E$
shows *group-epimorphism* (*galois-restriction* $K \ E \ F$)
 (*field-auto* $K \ F$) (*compose* K) (*identity* K)
 (*field-auto* $E \ F$) (*compose* E) (*identity* E)
 <proof>

context

fixes $K \ E \ F :: \text{complex set}$ **and** $p :: \text{complex poly}$
assumes *split:* *splitting-field* $F \ p \ K$
and *sfF:* *complex-subfield* F
and $E: E \in \text{inter-fields } K \ F$
and stable: $\bigwedge \sigma. \sigma \in \text{field-auto } K \ F \implies \sigma \restriction E = E$
begin

Now specialise to an intermediate field of a splitting field over the full base.

lemma *splitting-field-restriction-image:*

galois-restriction $K \ E \ F \restriction \text{field-auto } K \ F = \text{field-auto } E \ F$
 <proof>

lemma *splitting-field-restriction-kernel:*

group-homomorphism.Ker (*galois-restriction* $K \ E \ F$)
 (*field-auto* $K \ F$) (*identity* E) = *field-auto* $K \ E$
 <proof>

interpretation *restriction:* *group-epimorphism*

galois-restriction $K \ E \ F$
field-auto $K \ F$ *compose* K *identity* K
field-auto $E \ F$ *compose* E *identity* E
 <proof>

The first isomorphism theorem identifies the quotient by the relative automorphism group with the Galois group of the stable intermediate field.

theorem *splitting-field-restriction-factor-isomorphism:*

(*restriction.kernel.Factor-Group*,
 restriction.kernel.quotient-composition,
 restriction.kernel.Class (*identity* K)) $\cong_G$
 (*field-auto* $E \ F$, *compose* E , *identity* E)
 <proof>

For a finite splitting-field Galois group, the quotient isomorphism gives the expected multiplicative cardinality law. This is a group-order statement; identifying it with a general extension dimension is a separate finite-dimensional theorem.

theorem *splitting-field-restriction-card:*

card (*field-auto* $K \ F$) =
 card (*field-auto* $K \ E$) * *card* (*field-auto* $E \ F$)
 <proof>

end

end

104 Artin's lemma: a finite automorphism group bounds the degree over its fixed field

theory *Artin*

imports *Galois-Correspondence Linear-System Iso-Extension-Tower*
begin

Let H be a *finite* group of automorphisms of a subfield K of a field, and let *fixed-field* K^H be the subfield it fixes pointwise. *Artin's lemma* bounds the extension K over that fixed field by the order of the group: any $\text{card } H + 1$ elements of K are linearly dependent over *fixed-field* K^H .

The proof is the linear-algebra pigeonhole principle. Given elements $x_1, \dots, x_n$ of K with $n > \text{card } H$, consider the homogeneous system

- one equation per automorphism $\sigma \in H$, namely $\sum_j \sigma(x_j) c_j = 0$, in the unknowns c_j . There are more unknowns than equations, so *Subfield.underdetermined-solution* supplies a nontrivial solution in K — and the equation belonging to the identity automorphism is precisely the required dependence $\sum_j x_j c_j = 0$.

104.1 Artin's lemma

The system to be solved has one equation per automorphism and one unknown per element, so its coefficient matrix is $\lambda \sigma \ j. \ \sigma \ (x \ j)$: a genuinely two-index family, indexed by *functions* on one side and by the element index on the other. This is exactly the shape *Subfield.underdetermined-solution* was stated for, indices ranging over arbitrary finite sets rather than initial segments.

theorem *artin-lemma*:

fixes $H :: ('a :: \text{field} \Rightarrow 'a) \text{ set}$ **and** $x :: 'j \Rightarrow 'a$
assumes $K: \text{Subfield } K$
and $\text{fin}H: \text{finite } H$ **and** $HG: H \subseteq \text{field-auto } K$
and $\text{fin}J: \text{finite } J$ **and** $\text{card}: \text{card } H < \text{card } J$
and $xK: \bigwedge j. j \in J \implies x \ j \in K$
shows $\exists c. (\forall j. c \ j \in K) \wedge (\exists j \in J. c \ j \neq 0) \wedge (\forall \sigma \in H. (\sum j \in J. \sigma \ (x \ j) * c \ j) = 0)$
<proof>

The advertised form: more than $\text{card } H$ elements of K admit a nontrivial vanishing K -linear combination against *every* automorphism of H at once, and in particular — taking the identity, which lies in every Galois group — against the elements themselves.

corollary *artin-lemma-dependence*:

fixes $H :: ('a :: \text{field} \Rightarrow 'a) \text{ set}$ **and** $x :: 'j \Rightarrow 'a$
assumes $K: \text{Subfield } K$
and $\text{finH}: \text{finite } H$ **and** $\text{HG}: H \subseteq \text{field-auto } K \ F$ **and** $\text{id}: \text{identity } K \in H$
and $\text{finJ}: \text{finite } J$ **and** $\text{card}: \text{card } H < \text{card } J$
and $xJK: x \text{ ' } J \subseteq K$
shows $\exists c. (\forall j. c \ j \in K) \wedge (\exists j \in J. c \ j \neq 0) \wedge (\sum j \in J. x \ j * c \ j) = 0$
 $\langle \text{proof} \rangle$

For a genuine Galois subgroup the identity premise is automatic. This is the convenient interface for the later fixed-field argument, while *artin-lemma* deliberately retains the extra generality of an arbitrary finite automorphism family.

corollary *artin-lemma-subgroup-dependence:*

fixes $H :: ('a :: \text{field} \Rightarrow 'a) \text{ set}$ **and** $x :: 'j \Rightarrow 'a$
assumes $K: \text{Subfield } K$
and $H: H \in \text{galois-subgroups } K \ F$ **and** $\text{finH}: \text{finite } H$
and $\text{finJ}: \text{finite } J$ **and** $\text{card}: \text{card } H < \text{card } J$
and $xJK: x \text{ ' } J \subseteq K$
shows $\exists c. (\forall j. c \ j \in K) \wedge (\exists j \in J. c \ j \neq 0) \wedge (\sum j \in J. x \ j * c \ j) = 0$
 $\langle \text{proof} \rangle$

104.2 Transporting a dependence along an automorphism

The step that will drive the descent to the fixed field: an automorphism carries a vanishing linear combination to another vanishing linear combination, over the transported coefficients. Multiplicativity handles each summand, additivity the sum, and *field-auto-zero* the right-hand side.

This is the mechanism behind the minimality argument sketched at the end. If c is a dependence among the $x \ j$ with the fewest nonzero coefficients, normalised so that some $c \ k = 1$, then $\lambda j. \sigma \ (c \ j)$ is another dependence, also with $\sigma \ (c \ k) = 1$ and supported in the same places. Their difference is a shorter dependence, so minimality forces it to vanish, i.e. $\sigma \ (c \ j) = c \ j$ for every $\sigma \in H$ — which is exactly to say that the coefficients lie in *fixed-field* $K \ H$.

lemma *field-auto-transport-dependence:*

assumes $K: \text{Subfield } K$ **and** $s: \sigma \in \text{field-auto } K \ F$
and $\text{finJ}: \text{finite } J$
and $xK: \bigwedge j. j \in J \Rightarrow x \ j \in K$ **and** $cK: \bigwedge j. j \in J \Rightarrow c \ j \in K$
and $\text{dep}: (\sum j \in J. x \ j * c \ j) = 0$
shows $(\sum j \in J. \sigma \ (x \ j) * \sigma \ (c \ j)) = 0$
 $\langle \text{proof} \rangle$

Applying one subgroup automorphism to the coefficients preserves the simultaneous system. For a target row ϱ , apply σ to the row indexed by *compose* K (*inverse* σ) ϱ ; the group laws reduce the transformed row back to ϱ .

lemma *field-auto-transport-artin-solution:*

fixes $H :: ('a :: \text{field} \Rightarrow 'a) \text{ set}$
assumes $K: \text{Subfield } K$ **and** $H: H \in \text{galois-subgroups } K \ F$
and $s: \sigma \in H$ **and** $\text{finJ}: \text{finite } J$

and $xK: \bigwedge j. j \in J \implies x\ j \in K$
and $cK: \bigwedge j. j \in J \implies c\ j \in K$
and $equations: \forall \varrho \in H. (\sum j \in J. \varrho\ (x\ j) * c\ j) = 0$
and $\varrho \in H$
shows $(\sum j \in J. \varrho\ (x\ j) * \sigma\ (c\ j)) = 0$
 $\langle proof \rangle$

104.3 Descending the coefficients to the fixed field

The coefficients of a dependence can be taken to lie in the fixed field, not merely in K .

The device is minimality. Among all nontrivial dependences of the $x\ j$ choose one whose *support* $\{j \in J. c\ j \neq 0\}$ is smallest, and scale it so that some distinguished coefficient equals 1. For $\sigma \in H$ the transported family $\lambda j. \sigma\ (c\ j)$ is again a dependence — this is *field-auto-transport-dependence*, and it needs σ to fix the $x\ j$, which is why the $x\ j$ are taken in the fixed field's *ambient* position below.

lemma *dependence-difference*:

fixes $c\ d :: 'j \Rightarrow 'a :: field$
assumes $fnJ: finite\ J$
and $dep-c: (\sum j \in J. x\ j * c\ j) = 0$ **and** $dep-d: (\sum j \in J. x\ j * d\ j) = 0$
shows $(\sum j \in J. x\ j * (c\ j - d\ j)) = 0$
 $\langle proof \rangle$

The simultaneous system version of the minimal-support descent. Unlike the specialized fixed-vector descent below, this does not assume that the vectors are fixed by H : the equations contain every row $\varrho \in H$, and *field-auto-transport-artin-solution* transports a solution to another solution of that same system. This is the form needed for the genuine Artin bound.

theorem *artin-solution-over-fixed-field*:

fixes $H :: ('a :: field \Rightarrow 'a)\ set$ **and** $x :: 'j \Rightarrow 'a$
assumes $K: Subfield\ K$ **and** $H: H \in galois-subgroups\ K\ F$
and $fnJ: finite\ J$ **and** $xK: \bigwedge j. j \in J \implies x\ j \in K$
and $exsol: \exists c. (\forall j. c\ j \in K) \wedge (\exists j \in J. c\ j \neq 0) \wedge (\forall \varrho \in H. (\sum j \in J. \varrho\ (x\ j) * c\ j) = 0)$
shows $\exists c. (\forall j \in J. c\ j \in fixed-field\ K\ H) \wedge (\exists j \in J. c\ j \neq 0) \wedge (\forall \varrho \in H. (\sum j \in J. \varrho\ (x\ j) * c\ j) = 0)$
 $\langle proof \rangle$

Note that the $x\ j$ are *fixed* by every $\sigma \in H$. This is essential and not a technicality: *field-auto-transport-dependence* turns a dependence of the $x\ j$ into a dependence of the images $\sigma\ (x\ j)$, and only when those images are the $x\ j$ again is the result a dependence of the *same* family, so that subtracting it from the original stays inside the system. In the intended application the roles are reversed from what one might expect: it is the *vectors* that are taken over the fixed field and the *coefficients* that are unknown and must be shown to descend.

Also, H is closed under composition — a subgroup of the Galois group — which

is what makes $\sigma(cj) = cj$ for all $\sigma \in H$ equivalent to $cj \in \text{fixed-field } KH$. In fact only membership of $\sigma \in H \subseteq \text{field-auto } KF$ is used below; the group structure is what makes the conclusion *mean* what it says.

theorem *dependence-over-fixed-field:*

fixes $H :: ('a :: \text{field} \Rightarrow 'a) \text{ set}$ **and** $x :: 'j \Rightarrow 'a$
assumes $K: \text{Subfield } K$
and $HG: H \subseteq \text{field-auto } KF$
and $\text{fin}J: \text{finite } J$
and $xK: \bigwedge j. j \in J \Rightarrow xj \in K$
and $\text{xfix}: \bigwedge \sigma j. [\sigma \in H; j \in J] \Rightarrow \sigma(xj) = xj$
and $\text{exdep}: \exists c. (\forall j. cj \in K) \wedge (\exists j \in J. cj \neq 0) \wedge (\sum j \in J. xj * cj) = 0$
shows $\exists c. (\forall j \in J. cj \in \text{fixed-field } KH) \wedge (\exists j \in J. cj \neq 0)$
 $\wedge (\sum j \in J. xj * cj) = 0$

<proof>

104.4 Dedekind's lemma: distinct automorphisms are linearly independent

The reverse direction. Distinct field automorphisms are linearly independent as functions: if $\sum_{\sigma} a_{\sigma} \sigma(z) = 0$ for *every* z in K , with the σ pairwise distinct, then all the coefficients a_{σ} vanish.

The proof is by induction on the number of automorphisms involved, and the inductive step is a small piece of cunning. Suppose the relation holds for a set *insert* σ_0 S and, for a contradiction, that it is nontrivial. Pick τ in S and, since τ and σ_0 are distinct automorphisms, an element y of K with $\sigma_0 y \neq \tau y$. Now evaluate the relation at $y * z$ rather than at z : by multiplicativity this multiplies the σ -th term by σy . Subtracting $\sigma_0 y$ times the original relation annihilates the σ_0 term, leaving a relation on S alone whose τ -coefficient is $a_{\tau} * (\tau y - \sigma_0 y) \neq 0$.

lemma *character-relation-scaled:*

assumes $K: \text{Subfield } K$ **and** $\text{fin}S: \text{finite } S$
and $SG: S \subseteq \text{field-auto } KF$
and $\text{rel}: \bigwedge z. z \in K \Rightarrow (\sum \sigma \in S. a_{\sigma} * \sigma z) = 0$
and $y: y \in K$ **and** $z: z \in K$
shows $(\sum \sigma \in S. (a_{\sigma} * \sigma y) * \sigma z) = 0$

<proof>

Dedekind's lemma. The coefficients are arbitrary field elements; no subfield hypothesis on them is needed, and none is available in the intended application.

theorem *dedekind-independence:*

fixes $S :: ('a :: \text{field} \Rightarrow 'a) \text{ set}$ **and** $a :: ('a \Rightarrow 'a) \Rightarrow 'a$
assumes $K: \text{Subfield } K$
shows $[\text{finite } S; S \subseteq \text{field-auto } KF; \bigwedge z. z \in K \Rightarrow (\sum \sigma \in S. a_{\sigma} * \sigma z) = 0]$
 $\Rightarrow \forall \sigma \in S. a_{\sigma} = 0$

<proof>

This theory supplies the two difficult ingredients for Artin's theorem: the pi-

geonhole dependence and the descent of its coefficients into the fixed field. The vector-space degree assembly and the final finite-group equality live in *Artin-Degree*.

The older *dependence-over-fixed-field* remains a useful specialization for a family of vectors already fixed by H . The corrected simultaneous-system theorem *artin-solution-over-fixed-field* is the interface used by the degree layer, so the public *artin-theorem* no longer exposes either that fixed-vector premise or an explicit basis premise.

end

105 Artin's degree bound: the fixed field of a finite group has small index

theory *Artin-Degree*

imports *Artin Algebraic-Transitivity*

begin

The results of *Artin* are phrased with explicit sums over an index set. Here they are transported into the vector-space language of *Extension-Vector-Space*, where *fixed-field* $K \supseteq H \subseteq K$ is a *subfield-tower* and the size of the extension is measured by *Vector-Space.lin-indep* and *Vector-Space.dimension*. The bridge in both directions is *subfield-tower.vs-lincomb-eq-sum*: an abstract linear combination over a finite set of vectors is an ordinary field sum.

The conclusion is *Artin's bound*: over the fixed field of a finite group H of automorphisms, no more than $\text{card } H$ elements of K can be linearly independent. Together with Dedekind's lemma — which gives the matching lower bound — this is what pins the order of the Galois group of *fixed-field* $K \supseteq H$ to exactly $\text{card } H$.

notation *plus* (**infixl** $\langle + \rangle$ 65)

notation *minus* (**infixl** $\langle - \rangle$ 65)

unbundle *uminus-syntax*

105.1 The fixed field sits below K as a tower

Both fields are native set-based subfields and one contains the other, so they form a *subfield-tower* — the scalars being the fixed field and the vectors K .

lemma *fixed-field-tower*:

assumes K : *Subfield* K **and** HG : $H \subseteq \text{field-auto } K \ F$

shows *subfield-tower* (*fixed-field* $K \supseteq H$) K

<proof>

105.2 Artin's bound

No set of more than $\text{card } H$ elements of K is linearly independent over the fixed field. The argument is the one sketched in *Artin*, now assembled: *artin-lemma*

produces a simultaneous dependence with coefficients in K , and *artin-solution-over-fixed-field* pushes those coefficients down into the fixed field. The simultaneous system is the key correction: no assumption that the vectors themselves are fixed by H is needed.

Care is needed about which set indexes what. A *lin-indep* set is a set of *vectors*, and the dependence lemmas index by an arbitrary type; we index the vectors by themselves, taking $J = B$ and $x = id$.

The statements live in a *context* fixing K and H and interpreting the tower. The *Vector-Space* locale has nine parameters, so spelling *lin-indep* out at the top level would mean writing all of them; inside the context the interpretation supplies them and the qualified *T.vs.* prefix works.

context

fixes $K\ F :: 'a :: \text{field set}$ **and** $H :: ('a \Rightarrow 'a)\ \text{set}$
assumes K : *Subfield* K **and** $H\text{sub}$: $H \in \text{galois-subgroups } K\ F$

begin

lemma *H-auto*: $H \subseteq \text{field-auto } K\ F$ $\langle \text{proof} \rangle$

interpretation *T*: *subfield-tower fixed-field* $K\ H\ K$ $\langle \text{proof} \rangle$

theorem *artin-bound*:

assumes *finH*: *finite* H **and** *BK*: $B \subseteq K$
and *finB*: *finite* B **and** *lt*: $\text{card } H < \text{card } B$
shows $\neg T.\text{vs.lin-indep } B$

$\langle \text{proof} \rangle$

Consequently a basis of K over the fixed field — if there is one — has at most $\text{card } H$ elements, so the degree is bounded by the group order.

corollary *artin-dimension-le*:

assumes *finH*: *finite* H **and** *basis*: $T.\text{vs.basis } B$
shows $\text{card } B \leq \text{card } H$

$\langle \text{proof} \rangle$

And the same bound on the degree itself, since the dimension is the size of any basis (*Vector-Space.dimension-eq-any-field* — no finiteness of the base field needed).

corollary *artin-degree-le*:

assumes *finH*: *finite* H **and** *basis*: $T.\text{vs.basis } B$
shows $T.\text{vs.dimension} \leq \text{card } H$

$\langle \text{proof} \rangle$

end

105.3 The reverse bound: the Galois group is no larger than the degree

Now the other inequality, and the one that needs Dedekind's lemma. Let E be an intermediate field and B a basis of K over E . Then *field-auto* K E has at most *card* B elements.

The argument is again the linear pigeonhole, with the roles of the two index sets exchanged. Suppose there were more than *card* B automorphisms, and take a finite set S of them of size *card* $B + 1$. Consider the system with *one equation per basis vector* $v \in B$ and one unknown per automorphism: $\sum_{\sigma} a_{\sigma} \sigma(v) = 0$. It has more unknowns than equations, so *Subfield.underdetermined-solution* gives a nontrivial solution.

A solution of those finitely many equations already satisfies the corresponding relation at *every* point of K : an arbitrary $z \in K$ is an E -combination of B , and each automorphism is E -linear (it is additive, multiplicative, and fixes E), so the relation propagates from the basis to all of K by linearity. But then Dedekind's lemma forces every coefficient to vanish — contradicting nontriviality.

The propagation step is the only real work, and it is where the basis hypothesis is used.

As before the statements sit inside a context, so that the tower's interpretation supplies the nine parameters of *Vector-Space* and *basis* can be written qualified. The tower is *interpreted* here rather than passed as an assumption: a hypothesis *subfield-tower* E K would not bring *T.vs.basis* into scope for the statement itself.

context

fixes K $E :: 'a :: \text{field set}$

assumes K : *Subfield* K **and** E base: *subfield-tower* E K

begin

interpretation T : *subfield-tower* E K $\langle \text{proof} \rangle$

The base field lies inside the extension — the tower's own axiom.

lemma EK : $E \subseteq K$ $\langle \text{proof} \rangle$

An automorphism fixing E is E -linear on linear combinations of a finite set of vectors: it commutes with the sum, with the scalar products, and fixes the scalars.

lemma *field-auto-lincomb*:

assumes s : $\sigma \in \text{field-auto } K$ E **and** $\text{fin}B$: *finite* B **and** BK : $B \subseteq K$

and c : $\bigwedge v. v \in B \implies c \ v \in E$

shows $\sigma (\sum v \in B. c \ v * v) = (\sum v \in B. c \ v * \sigma \ v)$

$\langle \text{proof} \rangle$

Hence a relation among automorphisms that holds on a basis holds throughout K .

lemma *character-relation-from-basis*:

assumes *finS*: *finite S* **and** *SG*: $S \subseteq \text{field-auto } K \ E$
and *basis*: *T.vs.basis B*
and *rel*: $\forall v \in B. (\sum \sigma \in S. a \ \sigma * \sigma \ v) = 0$
and *z*: $z \in K$
shows $(\sum \sigma \in S. a \ \sigma * \sigma \ z) = 0$
<proof>

The reverse bound. With a basis of size *card B*, there cannot be more than *card B* distinct automorphisms of *K* fixing *E*.

theorem *galois-group-card-le*:
assumes *basis*: *T.vs.basis B*
and *finS*: *finite S* **and** *SG*: $S \subseteq \text{field-auto } K \ E$
shows *card S* $\leq$ *card B*
<proof>

end

105.4 Artin's theorem

The two bounds now meet. Let *H* be a finite group of automorphisms of *K*. The uniform bound first extracts a basis *B* of *K* over *fixed-field K H*; write *G* = *field-auto K (fixed-field K H)* for the Galois group of the fixed field.

- *artin-dimension-le* gives *card B* $\leq$ *card H*: were there more basis vectors than automorphisms, Artin's bound would make them dependent;
- *galois-group-card-le* gives *card G* $\leq$ *card B*: were there more automorphisms than basis vectors, Dedekind's lemma would be contradicted;
- *galois-extension.le-galois-group-fixed-field* — the unit of the adjunction, and the one part needing no hypotheses — gives $H \subseteq G$.

Chaining the first two bounds gives *card G* $\leq$ *card H*, and an inclusion between finite sets of equal cardinality is an equality. So the kernel operator of the Galois correspondence is the identity on finite subgroups: *every* automorphism fixing the fixed field of *H* already belongs to *H*.

This is the converse half of the correspondence, and the hypothesis that does the work is *finiteness of H*; no basis, separability, or normality is assumed by the public theorem.

Again a context, for the same reason as before: the locally extracted *basis* is a *Vector-Space* constant and can only be written qualified where an interpretation of the tower is in scope. Here the tower is the one built from *H* itself.

context
fixes *K F* :: '*a* :: *field set* **and** *H* :: ('*a* $\Rightarrow$ '*a*) *set*
assumes *K*: *Subfield K* **and** *Fsub*: *Subfield F* **and** *FK*: $F \subseteq K$
and *Hsub*: *H* $\in$ *galois-subgroups K F*
begin

lemma *HG*: $H \subseteq \text{field-auto } K \ F$ *<proof>*

interpretation *T*: *subfield-tower fixed-field* $K \ H \ K$
<proof>

theorem *artin-fixed-field-degree-data*:

assumes *finH*: *finite* H

shows $\exists B. \ T.\text{vs.basis } B \wedge \text{card } B = \text{card } H \wedge$
 $\text{finite } (\text{field-auto } K \ (\text{fixed-field } K \ H)) \wedge$
 $\text{card } (\text{field-auto } K \ (\text{fixed-field } K \ H)) = \text{card } H$

<proof>

theorem *artin-theorem*:

assumes *finH*: *finite* H

shows $\text{field-auto } K \ (\text{fixed-field } K \ H) = H$

<proof>

end

With *artin-theorem* the kernel operator of the Galois correspondence is the identity on finite subgroups. Together with the semi-inverse laws already in *Galois-Correspondence*, the correspondence restricts to a *bijection* between the finite subgroups of the Galois group and those intermediate fields that arise as fixed fields.

The classical finite correspondence is assembled downstream in *Galois-Finite-Correspondence*. There separability and normality show that every intermediate field is recovered from its relative automorphism group, while this theorem supplies the inverse law for every subgroup.

end

106 The fundamental theorem for finite Galois extensions

theory *Galois-Finite-Correspondence*

imports *Artin-Degree Galois-Degree Finite-Extension-Tower Galois-Restriction*

begin

Separability survives enlargement of the base field. The new minimal polynomial divides the old one, and root-squarefreeness passes to divisors.

theorem *finite-separable-extension-base-change*:

fixes $F \ E \ K :: 'a :: \text{field set}$

assumes *T*: *finite-subfield-tower* $F \ K$ **and** *sep*: *separable-extension* $K \ F$

and *sfE*: *Subfield* E **and** *FE*: $F \subseteq E$ **and** *EK*: $E \subseteq K$

shows *separable-extension* $K \ E$

<proof>

Normality also survives enlargement of the base inside a finite extension. The minimal polynomial over the larger base divides the old minimal polynomial, so every new conjugate is already an old conjugate.

theorem *finite-normal-extension-base-change:*

fixes $F E K :: 'a :: \text{field set}$

assumes $T: \text{finite-subfield-tower } F K$ **and** $\text{normal: normal-extension } K F$

and $\text{sfE: Subfield } E$ **and** $\text{FE: } F \subseteq E$ **and** $\text{EK: } E \subseteq K$

shows $\text{normal-extension } K E$

<proof>

106.1 The finite correspondence

context

fixes $F K :: 'a :: \text{alg-closed-field set}$

assumes $\text{finite: finite-subfield-tower } F K$

and $\text{normal: normal-extension } K F$

and $\text{separable: separable-extension } K F$

begin

interpretation $T: \text{finite-subfield-tower } F K$ *<proof>*

interpretation $GE: \text{galois-extension } K F$

<proof>

Any two conjugates over the base are related by an automorphism of a finite Galois extension. A primitive element presents the extension as the field generated by the roots of its minimal polynomial; multiplying that polynomial by the minimal polynomial lets the generic transitivity theorem act on both roots without changing the generated field.

theorem *finite-galois-root-transfer:*

assumes $aK: a \in K$ **and** $mb: \text{poly } (\text{minpoly } F a) b = 0$

shows $\exists \sigma \in \text{field-auto } K F. \sigma a = b$

<proof>

Every intermediate field is recovered as the fixed field of its relative Galois group. The upper extension remains finite, normal and separable. Its degree equals the order of the relative group; Artin's theorem computes the degree over that group's fixed field. The tower law then makes the intervening degree one.

theorem *finite-galois-fixed-field:*

assumes $E: E \in \text{inter-fields } K F$

shows $\text{fixed-field } K (\text{field-auto } K E) = E$

<proof>

Conversely, a subgroup of the finite Galois group is recovered from its fixed field by Artin's theorem.

theorem *finite-galois-fixed-group:*

assumes $H: H \in \text{galois-subgroups } K F$

shows $\text{field-auto } K (\text{fixed-field } K H) = H$

$\langle proof \rangle$

The two antitone maps are therefore mutually inverse bijections.

theorem *finite-galois-field-group-bij:*

bij-betw (field-auto K) (inter-fields K F) (galois-subgroups K F)

$\langle proof \rangle$

theorem *finite-galois-group-field-bij:*

bij-betw (fixed-field K) (galois-subgroups K F) (inter-fields K F)

$\langle proof \rangle$

theorem *finite-galois-correspondence:*

bij-betw (field-auto K) (inter-fields K F) (galois-subgroups K F) $\wedge$

bij-betw (fixed-field K) (galois-subgroups K F) (inter-fields K F)

$\langle proof \rangle$

106.2 Degree and index formulas

context

fixes *E* :: 'a set

assumes *intermediate*: *E* $\in$ *inter-fields K F*

begin

lemma *intermediate-data:*

Subfield E $\wedge$ F $\subseteq$ E $\wedge$ E $\subseteq$ K

$\langle proof \rangle$

lemma *TFE-raw: finite-subfield-tower F E*

$\langle proof \rangle$

lemma *TEK-raw: finite-subfield-tower E K*

$\langle proof \rangle$

interpretation *TFE: finite-subfield-tower F E* $\langle proof \rangle$

interpretation *TEK: finite-subfield-tower E K* $\langle proof \rangle$

lemma *normalEK: normal-extension K E*

$\langle proof \rangle$

lemma *separableEK: separable-extension K E*

$\langle proof \rangle$

lemma *relative-subgroup:*

Subgroup (field-auto K E) (field-auto K F) (compose K) (identity K)

$\langle proof \rangle$

interpretation *S: subgroup-of-group field-auto K E field-auto K F*

compose K identity K

$\langle proof \rangle$

The relative Galois group has order equal to the degree of the upper extension.

theorem *finite-galois-relative-group-order:*

$\text{card } (\text{field-auto } K \ E) = \text{TEK.extension-degree}$
 $\langle \text{proof} \rangle$

Its index in the full Galois group is the degree of the lower extension.

theorem *finite-galois-relative-group-index:*

$S.\text{index} = \text{TFE.extension-degree}$
 $\langle \text{proof} \rangle$

theorem *finite-galois-relative-degree-index:*

$\text{card } (\text{field-auto } K \ E) = \text{TEK.extension-degree} \wedge$
 $S.\text{index} = \text{TFE.extension-degree}$
 $\langle \text{proof} \rangle$

106.3 Normal intermediate fields

An intermediate field is normal over the base exactly when every automorphism of the ambient finite Galois extension preserves it setwise. Some duplication with a later proof.

theorem *finite-galois-normal-iff-stable:*

$\text{normal-extension } E \ F \longleftrightarrow$
 $(\forall \sigma \in \text{field-auto } K \ F. \sigma \restriction E = \text{id}_E)$
 $\langle \text{proof} \rangle$

Setwise stability is also exactly normality of the corresponding subgroup. A lot of duplication with the similar theorem below.

theorem *finite-galois-normal-subgroup-iff-stable:*

$\text{normal-subgroup } (\text{field-auto } K \ E) \ (\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K) \longleftrightarrow$
 $(\forall \sigma \in \text{field-auto } K \ F. \sigma \restriction E = \text{id}_E)$
 $\langle \text{proof} \rangle$

This is the normal-field/normal-subgroup clause of the fundamental theorem.

theorem *finite-galois-normal-iff-normal-subgroup:*

$\text{normal-extension } E \ F \longleftrightarrow$
 $\text{normal-subgroup } (\text{field-auto } K \ E) \ (\text{field-auto } K \ F) \ (\text{compose } K) \ (\text{identity } K)$
 $\langle \text{proof} \rangle$

106.4 Restriction and the quotient theorem

Restriction onto a stable intermediate field is surjective. A primitive element for the full extension supplies a finite root-generated presentation over the intermediate field, so the generic homomorphism-extension theorem applies.

theorem *finite-galois-restriction-image:*

assumes *stable:* $\bigwedge \sigma. \sigma \in \text{field-auto } K \ F \implies \sigma \restriction E = \text{id}_E$
shows *galois-restriction* $K \ E \ F \restriction \text{field-auto } K \ F = \text{field-auto } E \ F$

⟨proof⟩

context

assumes *normal-intermediate: normal-extension E F*

begin

lemma *stable-intermediate:*

$\bigwedge \sigma. \sigma \in \text{field-auto } K \ F \implies \sigma \restriction E = \text{id}_E$

⟨proof⟩

lemma *restriction-image:*

$\text{galois-restriction } K \ E \ F \restriction \text{field-auto } K \ F = \text{field-auto } E \ F$

⟨proof⟩

interpretation *restriction: group-epimorphism galois-restriction K E F*

field-auto K F compose K identity K

field-auto E F compose E identity E

⟨proof⟩

theorem *finite-galois-restriction-kernel: restriction.Ker = field-auto K E*

⟨proof⟩

The first isomorphism theorem gives $\text{Gal}(K/F) / \text{Gal}(K/E) \cong \text{Gal}(E/F)$; the preceding kernel theorem identifies the factor-group kernel with the relative Galois group literally.

theorem *finite-galois-quotient-isomorphism:*

(restriction.kernel.Factor-Group, restriction.kernel.quotient-composition, restriction.kernel.Class (identity K))

$\cong_G$

(field-auto E F, compose E, identity E)

⟨proof⟩

end

end

end

end

107 Radical extensions and solvability by radicals

theory *Radical-Extension*

imports *Simple-Extension*

begin

This theory begins the *forward* Abel–Ruffini direction (phase R1: the defini-

tions). It fixes the notion of *solvability by radicals* over a base field, on the same generic *'a/generate-field* foundation used for the quintic. Later phases will connect it to solvability of the Galois group (via the Galois correspondence, cyclotomic and Kummer theory). Here we only make the target statement expressible.

107.1 Radical steps and towers

A single *radical step* from F to F' adjoins an element a some power of which already lies in F : $F' = \text{generate-field } (F \cup \{a\})$ with $a^n \in F$ for some $0 < n$.

definition *radical-step* :: 'a :: field set $\Rightarrow$ 'a set $\Rightarrow$ bool
where *radical-step* $F F' \equiv (\exists a n. n > 0 \wedge a^n \in F \wedge F' = \text{generate-field } (F \cup \{a\}))$

A *radical tower* is a list of fields, each obtained from the previous by a radical step. *radical-tower* $F Fs$ says the tower starting at F and following Fs is valid (each consecutive pair is a radical step).

fun *radical-tower* :: 'a :: field set $\Rightarrow$ 'a set list $\Rightarrow$ bool **where**
radical-tower $F [] \longleftrightarrow \text{True}$
 $| \text{radical-tower } F (G \# Gs) \longleftrightarrow \text{radical-step } F G \wedge \text{radical-tower } G Gs$

The top field of a tower (the last field, or the base if the tower is empty).

definition *tower-top* :: 'a :: field set $\Rightarrow$ 'a set list $\Rightarrow$ 'a set
where *tower-top* $F Fs = \text{last } (F \# Fs)$

lemma *tower-top-Nil [simp]: tower-top* $F [] = F$
 $\langle \text{proof} \rangle$

lemma *tower-top-Cons [simp]: tower-top* $F (G \# Gs) = \text{tower-top } G Gs$
 $\langle \text{proof} \rangle$

107.2 Solvability by radicals

A set S (typically the roots of a polynomial, together with the base field) is *solvable by radicals* over F when there is a radical tower over F whose top field contains S .

definition *solvable-by-radicals* :: 'a :: field set $\Rightarrow$ 'a set $\Rightarrow$ bool
where *solvable-by-radicals* $F S \longleftrightarrow (\exists Fs. \text{radical-tower } F Fs \wedge S \subseteq \text{tower-top } F Fs)$

The base field itself is trivially solvable by radicals (the empty tower).

lemma *solvable-by-radicals-base:*
assumes $S \subseteq F$ **shows** *solvable-by-radicals* $F S$
 $\langle \text{proof} \rangle$

A single radical step gives a one-element tower.

lemma *solvable-by-radicals-step*:
assumes $n > 0$ **and** $a \wedge n \in F$ **and** $S \subseteq \text{generate-field } (F \cup \{a\})$
shows *solvable-by-radicals* $F S$
 $\langle \text{proof} \rangle$

107.3 Concatenation of radical towers

The top of a concatenated tower is the top of the second part (or of the first, if the second is empty).

lemma *tower-top-append*:
 $\text{tower-top } F (Fs @ Gs) = \text{tower-top } (\text{tower-top } F Fs) Gs$
 $\langle \text{proof} \rangle$

Radical towers concatenate. A radical tower over F followed by a radical tower over its top field $\text{tower-top } F Fs$ yields a radical tower over F along the appended list. This is the splicing primitive used to build a compositum of radical extensions into a single tower.

lemma *radical-tower-append*:
 $\text{radical-tower } F Fs \implies \text{radical-tower } (\text{tower-top } F Fs) Gs \implies \text{radical-tower } F (Fs @ Gs)$
 $\langle \text{proof} \rangle$

Iterated concatenation. Given a list of towers, each valid over the top reached so far, their concatenation is a single radical tower. We phrase validity of a list of towers recursively.

fun *radical-tower-list* :: $'a :: \text{field set} \Rightarrow 'a \text{ set list list} \Rightarrow \text{bool}$ **where**
 $\text{radical-tower-list } F [] \longleftrightarrow \text{True}$
 $| \text{radical-tower-list } F (Ts \# Tss) \longleftrightarrow$
 $\text{radical-tower } F Ts \wedge \text{radical-tower-list } (\text{tower-top } F Ts) Tss$

lemma *radical-tower-concat*:
 $\text{radical-tower-list } F Tss \implies \text{radical-tower } F (\text{concat } Tss)$
 $\langle \text{proof} \rangle$

lemma *tower-top-concat*:
 $\text{radical-tower-list } F Tss \implies \text{tower-top } F (\text{concat } Tss) = \text{foldl tower-top } F Tss$
 $\langle \text{proof} \rangle$

107.4 Base change: lifting a radical tower to a larger base

Adjoining a fixed set E to every field of a radical tower over F yields a radical tower over $\text{generate-field } (E \cup F)$: each step $H \rightsquigarrow \text{generate-field } (H \cup \{a\})$ becomes $\text{generate-field } (E \cup H) \rightsquigarrow \text{generate-field } (E \cup \text{generate-field } (H \cup \{a\}))$, which by the collapse lemmas is $\text{generate-field } (\text{generate-field } (E \cup H) \cup \{a\})$ with the same radical a (its power $a^n \in H \subseteq \text{generate-field } (E \cup H)$). This is what lets conjugate towers, each based at F , be spliced onto a growing compositum.

lemma *radical-step-basechange*:
assumes *radical-step* $F\ G$
shows *radical-step* (*generate-field* $(E \cup F)$) (*generate-field* $(E \cup G)$)
 $\langle \text{proof} \rangle$

Hence base change of a whole radical tower: mapping each field H to *generate-field* $(E \cup H)$ sends a radical tower over F to a radical tower over *generate-field* $(E \cup F)$.

lemma *radical-tower-basechange*:
 $\text{radical-tower } F\ Fs \implies \text{radical-tower } (\text{generate-field } (E \cup F))\ (\text{map } (\lambda H. \text{generate-field } (E \cup H))\ Fs)$
 $\langle \text{proof} \rangle$

end

108 Stability of root-generated intermediate fields

theory *Normal-Closure*
imports *Galois-Restriction Radical-Extension*
begin

The *solvable-tower* predicate of *Galois-Solvable-Tower* requires each intermediate field to be *stable* under the ambient Galois group. The reusable fact that supplies this is: a field *generate-field* $(F \cup X)$ generated over F by the *full* root set X of an F -polynomial is stable under every F -automorphism of any ambient field K containing it.

108.1 An automorphism fixes the base field setwise

An F -automorphism of K maps F onto F : it fixes F pointwise, and every element of F is the image of itself.

lemma *field-auto-image-base*:
assumes $s: \sigma \in \text{field-auto } K\ F$
shows $\sigma ` F = F$
 $\langle \text{proof} \rangle$

108.2 Root-generated fields are stable

The stability lemma. Let X be the full set of roots (in the ambient K) of an F -polynomial Phi , and let $E = \text{generate-field } (F \cup X)$ be the field it generates over F . Then E is stable under every F -automorphism of K .

theorem *root-generated-stable*:
fixes $K\ F :: \text{complex set}$ **and** $\text{Phi} :: \text{complex poly}$
assumes K : *Subfield* K **and** sfF : *Subfield* F **and** FK : $F \subseteq K$
and Phi : $\text{Phi} \in \text{poly-over } F$
and Xdef : $X = \{r. \text{poly } \text{Phi } r = 0\}$ **and** XK : $X \subseteq K$ **and** finX : *finite* X

and $s: \sigma \in \text{field-auto } K \ F$
shows $\sigma \text{ ' generate-field } (F \cup X) = \text{generate-field } (F \cup X)$
 <proof>

108.3 A field homomorphism carries radical steps to radical steps

A conjugate of a radical is a radical

lemma *radical-step-image*:
fixes $f :: \text{complex} \Rightarrow \text{complex}$
assumes $\text{hom}: \text{field-hom-on } K \ f$ **and** $\text{sfF}: \text{Subfield } F$ **and** $\text{FK}: F \subseteq K$
and $\text{fixF}: \bigwedge x. x \in F \implies f \ x = x$
and $\text{step}: \text{radical-step } F \ G$ **and** $\text{GK}: G \subseteq K$
shows $\text{radical-step } F \ (f \text{ ' } G)$
 <proof>

General form (no fixing hypothesis): a field homomorphism carries a radical step $F \rightsquigarrow G$ to the radical step $f \text{ ' } F \rightsquigarrow f \text{ ' } G$. This is the form that iterates along a tower, where after the first step the base is no longer fixed pointwise.

lemma *radical-step-image-gen*:
fixes $f :: \text{complex} \Rightarrow \text{complex}$
assumes $\text{hom}: \text{field-hom-on } K \ f$ **and** $\text{FK}: F \subseteq K$
and $\text{step}: \text{radical-step } F \ G$ **and** $\text{GK}: G \subseteq K$
shows $\text{radical-step } (f \text{ ' } F) \ (f \text{ ' } G)$
 <proof>

108.4 A field homomorphism carries radical towers to radical towers

Each field of a radical tower is a subfield (each is generated over the previous), and the tower is increasing. In particular every field is contained in any subfield K that contains the *top* field — but for the tower-image induction we only need that consecutive fields nest, which we package as: every field of the tower is a subfield of the last. We instead carry an explicit ambient K with *tower-top* F $Fs \subseteq K$.

lemma *radical-step-field*:
assumes $\text{radical-step } F \ G$ **shows** $\text{Subfield } G \wedge F \subseteq G$
 <proof>

lemma *radical-tower-subset-ambient*:
 $\text{Subfield } K \implies \text{Subfield } F \implies \text{radical-tower } F \ Fs \implies \text{tower-top } F \ Fs \subseteq K$
 $\implies F \subseteq K \wedge (\forall G \in \text{set } Fs. G \subseteq K)$
 <proof>

A conjugate of a radical tower is a radical tower. Applying a field homomorphism f to every field of a radical tower over F yields a radical tower over $f \text{ ' } F$: each step maps to a step by *radical-step-image-gen*.

theorem *radical-tower-image:*
fixes $f :: \text{complex} \Rightarrow \text{complex}$
assumes $K: \text{Subfield } K$ **and** $\text{hom}: \text{field-hom-on } K \text{ } f$
shows $\text{Subfield } F \Longrightarrow \text{radical-tower } F \text{ } Fs \Longrightarrow \text{tower-top } F \text{ } Fs \subseteq K$
 $\Longrightarrow \text{radical-tower } (f \text{ ` } F) \text{ } (\text{List.map } ((\text{ ` } f) \text{ } Fs)$
 $\langle \text{proof} \rangle$

108.5 The top of a radical tower is a subfield containing the base

lemma *subfield-tower-top:*
 $\text{Subfield } F \Longrightarrow \text{radical-tower } F \text{ } Fs \Longrightarrow \text{Subfield } (\text{tower-top } F \text{ } Fs) \wedge F \subseteq \text{tower-top } F \text{ } Fs$
 $\langle \text{proof} \rangle$

108.6 Splicing a list of towers over a common base into one radical tower

Base change of a tower onto a subfield containing the base. If F is a subfield of the subfield B , a radical tower over F base-changes to a radical tower over B (not merely over *generate-field* $(B \cup F)$): since B is a subfield containing F , *generate-field* $(B \cup F) = B$.

lemma *radical-tower-basechange-subfield:*
assumes $B: \text{Subfield } B$ **and** $FB: F \subseteq B$ **and** $\text{tower}: \text{radical-tower } F \text{ } Fs$
shows $\text{radical-tower } B \text{ } (\text{List.map } (\lambda H. \text{generate-field } (B \cup H)) \text{ } Fs)$
 $\langle \text{proof} \rangle$

Validity of a family of towers each based at F .

fun *all-towers-over* :: $'a :: \text{field set} \Rightarrow 'a \text{ set list list} \Rightarrow \text{bool}$ **where**
 $\text{all-towers-over } F \text{ } [] \longleftrightarrow \text{True}$
 $| \text{all-towers-over } F \text{ } (Ts \# Tss) \longleftrightarrow \text{radical-tower } F \text{ } Ts \wedge \text{all-towers-over } F \text{ } Tss$

Splice a family of towers (all based at F) into one list, base-changing each successive tower onto the running top B . The accumulator B is the subfield reached so far.

fun *spliced* :: $'a :: \text{field set} \Rightarrow 'a \text{ set list list} \Rightarrow 'a \text{ set list}$ **where**
 $\text{spliced } B \text{ } [] = []$
 $| \text{spliced } B \text{ } (Ts \# Tss) =$
 $(\text{let } Ts' = \text{List.map } (\lambda H. \text{generate-field } (B \cup H)) \text{ } Ts$
 $\text{in } Ts' @ \text{spliced } (\text{tower-top } B \text{ } Ts') \text{ } Tss)$

Splicing a family of towers-over- F onto a subfield B containing F yields a radical tower over B .

lemma *radical-tower-spliced:*
 $\text{Subfield } B \Longrightarrow \text{Subfield } F \Longrightarrow F \subseteq B \Longrightarrow \text{all-towers-over } F \text{ } Tss$
 $\Longrightarrow \text{radical-tower } B \text{ } (\text{spliced } B \text{ } Tss)$
 $\langle \text{proof} \rangle$

108.7 A radical tower is increasing; every field lies in the top

Each field of a radical tower over F is contained in the top field: the tower is increasing (each step is a field extension, *radical-step-field*), so F and every listed field embed into *tower-top* F F s.

lemma *field-subset-tower-top*:

Subfield $F \implies \text{radical-tower } F \text{ } F\text{s} \implies F \subseteq \text{tower-top } F \text{ } F\text{s} \wedge (\forall H \in \text{set } F\text{s}. H \subseteq \text{tower-top } F \text{ } F\text{s})$
 $\langle \text{proof} \rangle$

end

109 Finite field extensions

theory *Finite-Field-Extensions*

imports *Finite-Field-Frobenius Extension-Properties Algebraic-Transitivity*

begin

The polynomial $X^{\text{card } K} - X$ packages the two decisive properties of a finite subfield K : every element of the carrier is a root, and there can be no further roots. Its derivative is -1 , so minimal polynomials of elements of finite extensions divide a root-squarefree polynomial. This yields separability and normality directly in the native carrier-set representation.

definition *finite-field-vanishing-poly* :: $'a :: \text{field set} \Rightarrow 'a \text{ poly}$

where *finite-field-vanishing-poly* $K \equiv \text{monom } 1 (\text{card } K) - \text{monom } 1 1$

lemma *poly-finite-field-vanishing-poly* [*simp*]:

poly (*finite-field-vanishing-poly* K) $x = x^{\text{card } K} - x$

$\langle \text{proof} \rangle$

lemma *finite-field-vanishing-poly-nonzero*:

assumes $\text{card } K > 1$

shows *finite-field-vanishing-poly* $K \neq (0 :: 'a :: \text{field poly})$

$\langle \text{proof} \rangle$

lemma *degree-finite-field-vanishing-poly*:

assumes $\text{card } K > 1$

shows *degree* (*finite-field-vanishing-poly* $K :: 'a :: \text{field poly}$) = $\text{card } K$

$\langle \text{proof} \rangle$

Over any field, at most m elements are fixed by the m -power map when $m > 1$. This root bound later distinguishes the proper iterates of Frobenius.

lemma *card-power-fixed-points-le*:

fixes $m :: \text{nat}$

assumes $m: m > 1$

shows $\text{finite } \{x :: 'a :: \text{field}. x^m = x\} \wedge \text{card } \{x :: 'a. x^m = x\} \leq m$

<proof>

A root of multiplicity at least two is also a root of the formal derivative. Only this direction is characteristic-independent; it is the direction needed below.

lemma *order-ge-two-imp-pderiv-root*:
 fixes $p :: 'a :: \text{field poly}$
 assumes $p0: p \neq 0$ **and** $two: 2 \leq \text{order } a \ p$
 shows $\text{poly } (pderiv \ p) \ a = 0$
<proof>

context *Subfield*
begin

lemma *finite-field-vanishing-poly-over*:
 $\text{finite-field-vanishing-poly } K \in \text{poly-over } K$
<proof>

lemma *finite-field-vanishing-poly-roots*:
 assumes $\text{fin}: \text{finite } K$
 shows $\{x. \text{poly } (\text{finite-field-vanishing-poly } K) \ x = 0\} = K$
<proof>

lemma *finite-field-vanishing-poly-rsquarefree*:
 assumes $\text{fin}: \text{finite } K$
 shows $\text{rsquarefree } (\text{finite-field-vanishing-poly } K)$
<proof>

end

lemma *finite-field-vanishing-poly-over-subfield*:
 fixes $F \ K :: 'a :: \text{field set}$
 assumes $\text{sfF}: \text{Subfield } F$
 shows $\text{finite-field-vanishing-poly } K \in \text{poly-over } F$
<proof>

Every finite carrier extension is separable over each of its subfields. The minimal polynomial of an element of the top field divides the top field's vanishing polynomial, whose roots are all simple.

theorem *finite-subfield-extension-separable*:
 fixes $F \ K :: 'a :: \text{field set}$
 assumes $\text{sfF}: \text{Subfield } F$ **and** $\text{sfK}: \text{Subfield } K$ **and** $\text{FK}: F \subseteq K$
 and $\text{finK}: \text{finite } K$
 shows $\text{separable-extension } K \ F$
<proof>

Likewise every finite carrier extension is normal over each of its subfields. Any irreducible polynomial with one root in the top field has its minimal polynomial as an associate; that minimal polynomial divides the same vanishing polynomial, whose entire root set is the top carrier.

```

theorem finite-subfield-extension-normal:
  fixes  $F\ K :: 'a :: \text{field set}$ 
  assumes  $\text{sf}F$ : Subfield  $F$  and  $\text{sf}K$ : Subfield  $K$  and  $FK$ :  $F \subseteq K$ 
  and  $\text{fin}K$ : finite  $K$ 
  shows normal-extension  $K\ F$ 
   $\langle \text{proof} \rangle$ 

theorem finite-field-extension-cardinality:
  fixes  $F\ K :: 'a :: \text{field set}$ 
  assumes  $T$ : finite-subfield-tower  $F\ K$  and  $\text{fin}F$ : finite  $F$ 
  shows  $\text{card } K = \text{card } F \wedge \text{finite-subfield-tower.extension-degree } F\ K$ 
   $\langle \text{proof} \rangle$ 

theorem finite-field-extension-separable:
  fixes  $F\ K :: 'a :: \text{field set}$ 
  assumes  $T$ : finite-subfield-tower  $F\ K$  and  $\text{fin}F$ : finite  $F$ 
  shows separable-extension  $K\ F$ 
   $\langle \text{proof} \rangle$ 

theorem finite-field-extension-normal:
  fixes  $F\ K :: 'a :: \text{field set}$ 
  assumes  $T$ : finite-subfield-tower  $F\ K$  and  $\text{fin}F$ : finite  $F$ 
  shows normal-extension  $K\ F$ 
   $\langle \text{proof} \rangle$ 

end

```

110 Galois groups of finite fields

```

theory Finite-Field-Galois
  imports Finite-Field-Extensions Galois-Finite-Correspondence
begin

```

The restricted power map is a genuine element of the carrier-set automorphism group: outside the top carrier it has the canonical undefined value required by *PiE*.

```

definition finite-field-frobenius ::
   $'a :: \text{field set} \Rightarrow \text{nat} \Rightarrow ('a \Rightarrow 'a)$ 
  where finite-field-frobenius  $K\ q \equiv \text{restrict } (\text{frobenius-power } q)\ K$ 

```

```

lemma finite-field-frobenius-apply [simp]:
   $x \in K \Longrightarrow \text{finite-field-frobenius } K\ q\ x = x \wedge q$ 
   $\langle \text{proof} \rangle$ 

```

```

theorem finite-field-frobenius-in-field-auto:
  fixes  $F\ K :: 'a :: \text{field set}$ 
  assumes  $T$ : finite-subfield-tower  $F\ K$  and  $\text{fin}F$ : finite  $F$ 
  shows finite-field-frobenius  $K\ (\text{card } F) \in \text{field-auto } K\ F$ 

```

⟨proof⟩

theorem *finite-field-frobenius-power-apply*:

fixes $F K :: 'a :: \text{field set}$

assumes T : *finite-subfield-tower* $F K$ **and** finF : *finite* F **and** xK : $x \in K$

shows *Monoid.power* (*compose* K) (*identity* K)

(*finite-field-frobenius* K (*card* F)) i $x = x \wedge (\text{card } F \wedge i)$

⟨proof⟩

theorem *finite-field-frobenius-power-degree*:

fixes $F K :: 'a :: \text{alg-closed-field set}$

assumes T : *finite-subfield-tower* $F K$ **and** finF : *finite* F

and n : $n > 0$ *card* $K = \text{card } F \wedge n$

shows *Monoid.power* (*compose* K) (*identity* K)

(*finite-field-frobenius* K (*card* F)) $n = \text{identity } K$

⟨proof⟩

The Frobenius automorphism has exact order the degree of a finite extension. The upper bound is the finite-field identity on the top carrier; a smaller order would make every top-field element a root of a polynomial whose degree is too small.

theorem *finite-field-frobenius-order*:

fixes $F K :: 'a :: \text{alg-closed-field set}$

assumes T : *finite-subfield-tower* $F K$ **and** finF : *finite* F

shows *Monoid.element-order* (*compose* K) (*identity* K) (*finite-field-frobenius* K (*card* F)) =

finite-subfield-tower.extension-degree $F K$

⟨proof⟩

Every finite-field Galois group is generated by the relative Frobenius. The equality is proved at the carrier-set level, so downstream users can use the native automorphism group directly without introducing a second cyclic-group representation.

theorem *finite-field-galois-group-cyclic*:

fixes $F K :: 'a :: \text{alg-closed-field set}$

assumes T : *finite-subfield-tower* $F K$ **and** finF : *finite* F

shows *Group.cyclic-subgroup* (*compose* K) (*identity* K) (*finite-field-frobenius* K (*card* F)) = *field-auto* $K F$

⟨proof⟩

The canonical degree- d root set inside a finite extension. It is written as a set of ambient-field elements lying in the top carrier, which keeps the statement useful for the native set-based subfield representation.

definition *finite-field-power-roots* ::

$'a :: \text{field set} \Rightarrow 'a \text{ set} \Rightarrow \text{nat} \Rightarrow 'a \text{ set}$

where *finite-field-power-roots* $K F d \equiv \{x \in K. x \wedge (\text{card } F \wedge d) = x\}$

lemma *finite-field-power-roots-iff* [*simp*]:

$x \in \text{finite-field-power-roots } K F d \longleftrightarrow$
 $x \in K \wedge x^{\wedge (\text{card } F \wedge d)} = x$
 <proof>

Every intermediate field is exactly the root set of the appropriate Frobenius power. The root bound supplies the reverse inclusion: the intermediate field already has the full degree-many roots, so no additional ambient roots can occur.

theorem *finite-field-intermediate-power-roots:*
fixes $F K E :: 'a :: \text{alg-closed-field set}$
assumes $T: \text{finite-subfield-tower } F K$ **and** $\text{finF}: \text{finite } F$
and $E: E \in \text{inter-fields } K F$
shows $E = \text{finite-field-power-roots } K F (\text{finite-subfield-tower.extension-degree } F E)$
 <proof>

Consequently, the degree is a complete invariant for intermediate fields of a finite field extension.

theorem *finite-field-intermediate-degree-injective:*
fixes $F K :: 'a :: \text{alg-closed-field set}$
assumes $T: \text{finite-subfield-tower } F K$ **and** $\text{finF}: \text{finite } F$
and $E: E \in \text{inter-fields } K F$ **and** $E': E' \in \text{inter-fields } K F$
and $\text{degree-eq}: \text{finite-subfield-tower.extension-degree } F E = \text{finite-subfield-tower.extension-degree } F E'$
shows $E = E'$
 <proof>

The lower degree of every intermediate field divides the total extension degree, as expected from the finite-field divisor classification.

theorem *finite-field-intermediate-degree-dvd:*
fixes $F K E :: 'a :: \text{alg-closed-field set}$
assumes $T: \text{finite-subfield-tower } F K$ **and** $\text{finF}: \text{finite } F$
and $E: E \in \text{inter-fields } K F$
shows $\text{finite-subfield-tower.extension-degree } F E \text{ dvd } \text{finite-subfield-tower.extension-degree } F K$
 <proof>

Every divisor of the total degree is realised by a unique intermediate field. The subgroup generated by the corresponding power of Frobenius has the complementary order; finite Galois correspondence and the tower law then compute the degree of its fixed field.

theorem *finite-field-intermediate-exists:*
fixes $F K :: 'a :: \text{alg-closed-field set}$
assumes $T: \text{finite-subfield-tower } F K$ **and** $\text{finF}: \text{finite } F$
and $d: d \text{ dvd } \text{finite-subfield-tower.extension-degree } F K$
shows $\exists E. E \in \text{inter-fields } K F \wedge \text{finite-subfield-tower.extension-degree } F E = d$
 $\wedge E = \text{finite-field-power-roots } K F d$
 <proof>

end

111 The New-Algebra Development

theory *New-Algebra*

imports

Subgroup-Lattice
Monoid-Iso-Classes
Normal-Series
Normal-Chain
Group-Product
Group-Family-Product
Schreier-Refinement
Simple-Factor-Chain
Jordan-Hoelder-Uniqueness
Maximal-Normal-Subgroup
Finite-Composition-Series
Sn-Composition-Factors
Abel-Ruffini
P-Group

Subring-Generated
Finite-Field-Frobenius
Ideal-Extension
Chinese-Remainder-Rings
Field-Of-Fractions
Poly-Divisibility
Closure-Algebraic

Module-Complements
Module-Exact-Sequence
Module-Family-Product
Free-Module-Universal
Finite-Dimensional-Isomorphism
Finite-Dimensional-Splitting

Field-Extension
Field-Mult-Cyclic
Galois-Restriction-Splitting
Galois-Finite-Correspondence
Normal-Closure

Finite-Field-Galois
GF-p

begin

end

References

- [1] C. Ballarin. A case study in basic algebra. *Archive of Formal Proofs*, 2019. https://isa-afp.org/entries/Jacobson_Basic_Algebra.html, Formal proof development.
- [2] N. Jacobson. *Basic Algebra I*. W. H. Freeman, second edition, 1985.