

---

# Contents

<b>1</b>	<b>A Brief Overview</b> .....	1
1.1	Expressions, Types, and Functions .....	2
1.2	Propositions and Proofs .....	3
1.3	Propositions and Types .....	4
1.4	Specifications and Certified Programs .....	5
1.5	A Sorting Example .....	5
1.5.1	Inductive Definitions .....	6
1.5.2	The Relation “to have the same elements” .....	6
1.5.3	A Specification for a Sorting Program .....	7
1.5.4	An Auxiliary Function .....	7
1.5.5	The Main Sorting Function .....	8
1.6	Learning <i>Coq</i> .....	9
1.7	Contents of This Book .....	9
1.8	Lexical Conventions .....	11
<b>2</b>	<b>Types and Expressions</b> .....	13
2.1	First Steps .....	13
2.1.1	Terms, Expressions, Types .....	14
2.1.2	Interpretation Scopes .....	14
2.1.3	Type Checking .....	15
2.2	The Rules of the Game .....	17
2.2.1	Simple Types .....	17
2.2.2	Identifiers, Environments, Contexts .....	18
2.2.3	Expressions and Their Types .....	20
2.2.4	Free and Bound Variables; $\alpha$ -conversion .....	27
2.3	Declarations and Definitions .....	29
2.3.1	Global Declarations and Definitions .....	29
2.3.2	Sections and Local Variables .....	30
2.4	Computing .....	33
2.4.1	Substitution .....	34
2.4.2	Reduction Rules .....	34

2.4.3	Reduction Sequences	36
2.4.4	Convertibility	37
2.5	Types, Sorts, and Universes	37
2.5.1	The <code>Set</code> Sort	37
2.5.2	Universes	38
2.5.3	Definitions and Declarations of Specifications	39
2.6	Realizing Specifications	41
<b>3</b>	<b>Propositions and Proofs</b>	<b>43</b>
3.1	Minimal Propositional Logic	45
3.1.1	The World of Propositions and Proofs	45
3.1.2	Goals and Tactics	46
3.1.3	A First Goal-directed Proof	47
3.2	Relating Typing Rules and Tactics	51
3.2.1	Proposition Building Rules	51
3.2.2	Inference Rules and Tactics	52
3.3	Structure of an Interactive Proof	56
3.3.1	Activating the Goal Handling System	57
3.3.2	Current Stage of an Interactive Proof	57
3.3.3	Undoing	57
3.3.4	Regular End of a Proof	58
3.4	Proof Irrelevance	58
3.4.1	<code>Theorem</code> Versus <code>Definition</code>	59
3.4.2	Are Tactics Helpful for Building Programs?	59
3.5	Sections and Proofs	60
3.6	Composing Tactics	61
3.6.1	Tacticals	61
3.6.2	Maintenance Issues	65
3.7	On Completeness for Propositional Logic	67
3.7.1	A Complete Set of Tactics	67
3.7.2	Unprovable Propositions	68
3.8	Some More Tactics	68
3.8.1	The <code>cut</code> and <code>assert</code> Tactics	68
3.8.2	An Introduction to Automatic Tactics	70
3.9	A New Kind of Abstraction	71
<b>4</b>	<b>Dependent Products, or Pandora's Box</b>	<b>73</b>
4.1	In Praise of Dependence	74
4.1.1	Extending the Scope of Arrows	74
4.1.2	On Binding	78
4.1.3	A New Construct	79
4.2	Typing Rules and Dependent Products	81
4.2.1	The Application Typing Rule	81
4.2.2	The Abstraction Typing Rule	84
4.2.3	Type Inference	86

4.2.4	The Conversion Rule .....	90
4.2.5	Dependent Products and the Convertibility Order .....	90
4.3	* Expressive Power of the Dependent Product .....	91
4.3.1	Formation Rule for Products .....	91
4.3.2	Dependent Types .....	92
4.3.3	Polymorphism .....	94
4.3.4	Equality in the <i>Coq</i> System .....	98
4.3.5	Higher-Order Types .....	99
<b>5</b>	<b>Everyday Logic</b> .....	<b>105</b>
5.1	Practical Aspects of Dependent Products .....	105
5.1.1	<code>exact</code> and <code>assumption</code> .....	105
5.1.2	The <code>intro</code> Tactic .....	106
5.1.3	The <code>apply</code> Tactic .....	108
5.1.4	The <code>unfold</code> Tactic .....	115
5.2	Logical Connectives .....	116
5.2.1	Introduction and Elimination Rules .....	116
5.2.2	Using Contradictions .....	118
5.2.3	Negation .....	119
5.2.4	Conjunction and Disjunction .....	121
5.2.5	About the <code>repeat</code> Tactical .....	123
5.2.6	Existential Quantification .....	123
5.3	Equality and Rewriting .....	124
5.3.1	Proving Equalities .....	124
5.3.2	Using Equality: Rewriting Tactics .....	125
5.3.3	* The <code>pattern</code> Tactic .....	127
5.3.4	* Conditional Rewriting .....	128
5.3.5	Searching Theorems for Rewriting .....	129
5.3.6	Other Tactics on Equality .....	129
5.4	Tactic Summary Table .....	130
5.5	*** Impredicative Definitions .....	130
5.5.1	Warning .....	130
5.5.2	True and False .....	130
5.5.3	Leibniz Equality .....	131
5.5.4	Some Other Connectives and Quantifiers .....	133
5.5.5	A Guideline for Impredicative Definitions .....	135
<b>6</b>	<b>Inductive Data Types</b> .....	<b>137</b>
6.1	Types Without Recursion .....	137
6.1.1	Enumerated Types .....	137
6.1.2	Simple Reasoning and Computing .....	139
6.1.3	The <code>elim</code> Tactic .....	141
6.1.4	Pattern Matching .....	142
6.1.5	Record Types .....	145
6.1.6	Records with Variants .....	146

6.2	Case-Based Reasoning	148
6.2.1	The <code>case</code> Tactic	148
6.2.2	Contradictory Equalities	151
6.2.3	Injective Constructors	153
6.2.4	Inductive Types and Equality	156
6.2.5	* Guidelines for the <code>case</code> Tactic	156
6.3	Recursive Types	160
6.3.1	Natural Numbers as an Inductive Type	161
6.3.2	Proof by Induction on Natural Numbers	162
6.3.3	Recursive Programming	164
6.3.4	Variations in the Form of Constructors	167
6.3.5	** Types with Functional Fields	170
6.3.6	Proofs on Recursive Functions	172
6.3.7	Anonymous Recursive Functions ( <code>fix</code> )	174
6.4	Polymorphic Types	175
6.4.1	Polymorphic Lists	175
6.4.2	The <code>option</code> Type	177
6.4.3	The Type of Pairs	179
6.4.4	The Type of Disjoint Sums	180
6.5	* Dependent Inductive Types	180
6.5.1	First-Order Data as Parameters	180
6.5.2	Variably Dependent Inductive Types	181
6.6	* Empty Types	184
6.6.1	Non-dependent Empty Types	184
6.6.2	Dependence and Empty Types	185
7	<b>Tactics and Automation</b>	187
7.1	Tactics for Inductive Types	187
7.1.1	Case-by-Case Analysis and Recursion	187
7.1.2	Conversions	188
7.2	Tactics <code>auto</code> and <code>eauto</code>	190
7.2.1	Tactic Database Handling: <code>hint</code>	191
7.2.2	* The <code>eauto</code> Tactic	194
7.3	Automatic Tactics for Rewriting	194
7.3.1	The <code>autorewrite</code> Tactic	194
7.3.2	The <code>subst</code> Tactic	195
7.4	Numerical Tactics	196
7.4.1	The <code>ring</code> Tactic	196
7.4.2	The <code>omega</code> Tactic	198
7.4.3	The <code>field</code> Tactic	199
7.4.4	The <code>fourier</code> Tactic	200
7.5	Other Decision Procedures	200
7.6	** The Tactic Definition Language	201
7.6.1	Arguments in Tactics	202
7.6.2	Pattern Matching	203

7.6.3	Using Reduction in Defined Tactics	210
<b>8</b>	<b>Inductive Predicates</b>	211
8.1	Inductive Properties	211
8.1.1	A Few Examples	211
8.1.2	Inductive Predicates and Logic Programming	213
8.1.3	Advice for Inductive Definitions	214
8.1.4	The Example of Sorted Lists	215
8.2	Inductive Properties and Logical Connectives	217
8.2.1	Representing Truth	218
8.2.2	Representing Contradiction	218
8.2.3	Representing Conjunction	219
8.2.4	Representing Disjunction	219
8.2.5	Representing Existential Quantification	219
8.2.6	Representing Equality	220
8.2.7	*** Heterogeneous Equality	220
8.2.8	An Exotic Induction Principle?	225
8.3	Reasoning about Inductive Properties	226
8.3.1	Structured <code>intros</code>	226
8.3.2	The <code>constructor</code> Tactics	227
8.3.3	* Induction on Inductive Predicates	227
8.3.4	* Induction on <code>le</code>	229
8.4	* Inductive Relations and Functions	233
8.4.1	Representing the Factorial Function	234
8.4.2	** Representing the Semantics of a Language	239
8.4.3	** Proving Semantic Properties	240
8.5	* Elaborate Behavior of <code>elim</code>	244
8.5.1	Instantiating the Argument	244
8.5.2	Inversion	246
<b>9</b>	<b>* Functions and Their Specifications</b>	251
9.1	Inductive Types for Specifications	252
9.1.1	The “Subset” Type	252
9.1.2	Nested Subset Types	254
9.1.3	Certified Disjoint Sum	254
9.1.4	Hybrid Disjoint Sum	256
9.2	Strong Specifications	256
9.2.1	Well-specified Functions	257
9.2.2	Building Functions as Proofs	257
9.2.3	Preconditions for Partial Functions	258
9.2.4	** Proving Preconditions	259
9.2.5	** Reinforcing Specifications	260
9.2.6	*** Minimal Specification Strengthening	261
9.2.7	The <code>refine</code> Tactic	265
9.3	Variations on Structural Recursion	267

9.3.1	Structural Recursion with Multiple Steps . . . . .	267
9.3.2	Simplifying the Step . . . . .	271
9.3.3	Recursive Functions with Several Arguments . . . . .	271
9.4	** Binary Division . . . . .	276
9.4.1	Weakly Specified Division . . . . .	276
9.4.2	Well-specified Binary Division . . . . .	281
<b>10</b>	<b>* Extraction and Imperative Programming . . . . .</b>	<b>285</b>
10.1	Extracting Toward Functional Languages . . . . .	285
10.1.1	The Extraction Command . . . . .	286
10.1.2	The Extraction Mechanism . . . . .	287
10.1.3	Prop, Set, and Extraction . . . . .	295
10.2	Describing Imperative Programs . . . . .	297
10.2.1	The <i>Why</i> Tool . . . . .	297
10.2.2	*** The Inner Workings of <i>Why</i> . . . . .	300
<b>11</b>	<b>* A Case Study . . . . .</b>	<b>309</b>
11.1	Binary Search Trees . . . . .	309
11.1.1	The Data Structure . . . . .	309
11.1.2	A Naïve Approach to Deciding Occurrence . . . . .	311
11.1.3	Describing Search Trees . . . . .	311
11.2	Specifying Programs . . . . .	313
11.2.1	Finding an Occurrence . . . . .	313
11.2.2	Inserting a Number . . . . .	313
11.2.3	** Removing a Number . . . . .	314
11.3	Auxiliary Lemmas . . . . .	315
11.4	Realizing Specifications . . . . .	315
11.4.1	Realizing the Occurrence Test . . . . .	315
11.4.2	Insertion . . . . .	318
11.4.3	Removing Elements . . . . .	322
11.5	Possible Improvements . . . . .	323
11.6	Another Example . . . . .	324
<b>12</b>	<b>* The Module System . . . . .</b>	<b>325</b>
12.1	Signatures . . . . .	326
12.2	Modules . . . . .	328
12.2.1	Building a Module . . . . .	328
12.2.2	An Example: Keys . . . . .	329
12.2.3	Parametric Modules (Functors) . . . . .	332
12.3	A Theory of Decidable Order Relations . . . . .	335
12.3.1	Enriching a Theory with a Functor . . . . .	335
12.3.2	Lexicographic Order as a Functor . . . . .	337
12.4	A Dictionary Module . . . . .	339
12.4.1	Enriched Implementations . . . . .	340
12.4.2	Constructing a Dictionary with a Functor . . . . .	340

12.4.3	A Trivial Implementation	340
12.4.4	An Efficient Implementation	342
12.5	Conclusion	345
<b>13</b>	<b>** Infinite Objects and Proofs</b>	<b>347</b>
13.1	Co-inductive Types	347
13.1.1	The <code>CoInductive</code> Command	347
13.1.2	Specific Features of Co-inductive Types	348
13.1.3	Infinite Lists (Streams)	348
13.1.4	Lazy Lists	349
13.1.5	Lazy Binary Trees	349
13.2	Techniques for Co-inductive Types	350
13.2.1	Building Finite Objects	350
13.2.2	Pattern Matching	350
13.3	Building Infinite Objects	351
13.3.1	A Failed Attempt	352
13.3.2	The <code>CoFixpoint</code> Command	352
13.3.3	A Few Co-recursive Functions	354
13.3.4	Badly Formed Definitions	356
13.4	Unfolding Techniques	357
13.4.1	Systematic Decomposition	358
13.4.2	Applying the Decomposition Lemma	358
13.4.3	Simplifying a Call to a Co-recursive Function	359
13.5	Inductive Predicates over Co-inductive Types	361
13.6	Co-inductive Predicates	362
13.6.1	A Predicate for Infinite Sequences	363
13.6.2	Building Infinite Proofs	363
13.6.3	Guard Condition Violation	365
13.6.4	Elimination Techniques	366
13.7	Bisimilarity	368
13.7.1	The <code>bisimilar</code> Predicate	368
13.7.2	Using Bisimilarity	370
13.8	The Park Principle	371
13.9	LTL	372
13.10A	Case Study: Transition Systems	375
13.10.1	Automata and Traces	375
13.11	Conclusion	376
<b>14</b>	<b>** Foundations of Inductive Types</b>	<b>377</b>
14.1	Formation Rules	377
14.1.1	The Inductive Type	377
14.1.2	The Constructors	379
14.1.3	Building the Induction Principle	382
14.1.4	Typing Recursors	385
14.1.5	Induction Principles for Predicates	392

14.1.6	The <code>Scheme</code> Command	394
14.2	*** Pattern Matching and Recursion on Proofs	394
14.2.1	Restrictions on Pattern Matching	395
14.2.2	Relaxing the Restrictions	396
14.2.3	Recursion	398
14.3	Mutually Inductive Types	400
14.3.1	Trees and Forests	400
14.3.2	Proofs by Mutual Induction	402
14.3.3	*** Trees and Tree Lists	404
<b>15</b>	<b>* General Recursion</b>	<b>407</b>
15.1	Bounded Recursion	408
15.2	** Well-founded Recursive Functions	411
15.2.1	Well-founded Relations	411
15.2.2	Accessibility Proofs	411
15.2.3	Assembling Well-founded Relations	413
15.2.4	Well-founded Recursion	414
15.2.5	The Recursor <code>well_founded_induction</code>	414
15.2.6	Well-founded Euclidean Division	415
15.2.7	Nested Recursion	419
15.3	** General Recursion by Iteration	420
15.3.1	The Functional Related to a Recursive Function	421
15.3.2	Termination Proof	421
15.3.3	Building the Actual Function	424
15.3.4	Proving the Fixpoint Equation	424
15.3.5	Using the Fixpoint Equation	426
15.3.6	Discussion	427
15.4	*** Recursion on an Ad Hoc Predicate	427
15.4.1	Defining an Ad Hoc Predicate	428
15.4.2	Inversion Theorems	428
15.4.3	Defining the Function	429
15.4.4	Proving Properties of the Function	430
<b>16</b>	<b>* Proof by Reflection</b>	<b>433</b>
16.1	General Presentation	433
16.2	Direct Computation Proofs	435
16.3	** Proof by Algebraic Computation	438
16.3.1	Proofs Modulo Associativity	438
16.3.2	Making the Type and the Operator More Generic	442
16.3.3	*** Commutativity: Sorting Variables	445
16.4	Conclusion	447
<b>Appendix</b>		<b>449</b>
Insertion Sort		449



<b>References</b> .....	453
<b>Index</b> .....	459
Coq and Its Libraries .....	460
Examples from the Book .....	464



<http://www.springer.com/978-3-540-20854-9>

Interactive Theorem Proving and Program Development

Coq'Art: The Calculus of Inductive Constructions

Bertot, Y.; Castéran, P.

2004, XXV, 472 p. 1 illus., Hardcover

ISBN: 978-3-540-20854-9