
Contents

| | | |
|----------------|--|----|
| Preface | | xv |
| I | INTRODUCTION | 1 |
| 1 | An Overview of Fundamental Proof Methods | 3 |
| | 1.1 Equality chaining | 3 |
| | 1.2 Induction | 8 |
| | 1.3 Case analysis | 10 |
| | 1.4 Proof by contradiction | 12 |
| | 1.5 Abstraction/specialization | 13 |
| | 1.6 The usual case: Proof methods in combination | 15 |
| | 1.7 Automated proof | 15 |
| | 1.8 Structure of the book | 16 |
| 2 | Introduction to Athena | 19 |
| | 2.1 Interacting with Athena | 22 |
| | 2.2 Domains and function symbols | 23 |
| | 2.3 Terms | 27 |
| | 2.4 Sentences | 34 |
| | 2.5 Definitions | 40 |
| | 2.6 Assumption bases | 42 |
| | 2.7 Datatypes | 44 |
| | 2.8 Polymorphism | 50 |
| | 2.8.1 Polymorphic domains and sort identity | 50 |
| | 2.8.2 Polymorphic function symbols | 52 |
| | 2.8.3 Polymorphic datatypes | 57 |
| | 2.8.4 Integers and reals | 59 |
| | 2.9 Meta-identifiers | 61 |
| | 2.10 Expressions and deductions | 62 |
| | 2.10.1 Compositions | 65 |
| | 2.10.2 Nested method calls | 68 |
| | 2.10.3 Let expressions and deductions | 69 |
| | 2.10.4 Conclusion-annotated deductions | 70 |
| | 2.10.5 Conditional expressions and deductions | 71 |
| | 2.10.6 Pattern-matching expressions and deductions | 72 |
| | 2.10.7 Backtracking expressions and deductions | 73 |
| | 2.10.8 Defining procedures and methods | 74 |

| | | |
|-----------|---|------------|
| 2.11 | More on pattern matching | 78 |
| 2.12 | Directives | 85 |
| 2.13 | Overloading | 86 |
| 2.14 | Programming | 89 |
| | 2.14.1 Characters | 90 |
| | 2.14.2 Strings | 90 |
| | 2.14.3 Cells and vectors | 90 |
| | 2.14.4 Tables and maps | 91 |
| | 2.14.5 While loops | 93 |
| | 2.14.6 Expression sequences | 93 |
| | 2.14.7 Recursion | 93 |
| | 2.14.8 Substitutions | 94 |
| 2.15 | A consequence of static scoping | 98 |
| 2.16 | Miscellanea | 99 |
| 2.17 | Summary and notational conventions | 103 |
| 2.18 | Exercises | 105 |
| II | FUNDAMENTAL PROOF METHODS | 111 |
| 3 | Proving Equalities | 113 |
| 3.1 | Numeric equations | 113 |
| 3.2 | Equality chaining preview | 116 |
| 3.3 | Terms and sentences as trees | 117 |
| 3.4 | The logic behind equality chaining | 120 |
| 3.5 | More examples of equality chaining | 128 |
| 3.6 | A more substantial proof example | 130 |
| 3.7 | A better proof | 134 |
| 3.8 | The principle of mathematical induction | 135 |
| | 3.8.1 Different ways of understanding mathematical induction | 139 |
| 3.9 | List equations | 140 |
| | 3.9.1 Polymorphic datatypes | 147 |
| 3.10 | Evaluation of ground terms | 150 |
| 3.11 | Top-down proof development | 152 |
| 3.12 | ★ Input expansion and output transformation | 158 |
| | 3.12.1 Converters | 158 |
| | 3.12.2 Input expansion | 162 |
| | 3.12.3 Output transformation | 165 |
| | 3.12.4 Combining input expansion and output transformation with overloading | 166 |
| | 3.12.5 Using <code>declare</code> with auxiliary information | 167 |

CONTENTS

vii

| | | |
|----------|--|------------|
| 3.13 | ★ Conjecture falsification | 168 |
| 3.14 | ★ Conditional rewriting and additional chaining features | 172 |
| 3.15 | ★ Proper function definitions | 179 |
| 3.16 | Summary | 185 |
| 3.17 | Additional exercises | 186 |
| 3.18 | Chapter notes | 189 |
| 4 | Sentential Logic | 191 |
| 4.1 | Working with the Boolean constants | 191 |
| 4.2 | Working with conjunctions | 192 |
| 4.2.1 | Using conjunctions: left-and and right-and | 192 |
| 4.2.2 | Deriving conjunctions: both | 193 |
| 4.3 | Working with conditionals | 194 |
| 4.3.1 | Using conditionals: modus ponens and modus tollens | 194 |
| 4.3.2 | Deriving conditionals: The assume construct | 195 |
| 4.4 | Working with disjunctions | 199 |
| 4.4.1 | Using disjunctions: Reasoning by cases | 199 |
| 4.4.2 | Deriving disjunctions | 201 |
| 4.5 | Working with negations | 202 |
| 4.5.1 | Using negations | 202 |
| 4.5.2 | Deriving negations: Proof by contradiction | 202 |
| 4.6 | Working with biconditionals | 206 |
| 4.6.1 | Using biconditionals | 206 |
| 4.6.2 | Deriving biconditionals | 207 |
| 4.7 | Forcing a proof | 207 |
| 4.8 | Putting it all together | 209 |
| 4.9 | A library of useful methods for sentential reasoning | 211 |
| 4.10 | Recursive proof methods | 226 |
| 4.11 | Dealing with large conjunctions and disjunctions | 236 |
| 4.12 | Sentential logic semantics | 238 |
| 4.13 | SAT solving | 246 |
| 4.14 | Proof heuristics for sentential logic | 272 |
| 4.14.1 | Backward tactics | 274 |
| 4.14.2 | Forward tactics | 276 |
| 4.14.3 | Replacement tactics | 282 |
| 4.14.4 | Strategies for deploying the tactics | 282 |
| 4.15 | A theorem prover for sentential logic (optional) | 294 |
| 4.16 | Additional exercises | 304 |
| 4.17 | Chapter notes | 315 |

| | | |
|------------|---|-----|
| 5 | First-Order Logic | 319 |
| | 5.1 Working with universal quantifications | 323 |
| | 5.1.1 Using universal quantifications | 323 |
| | 5.1.2 Deriving universal quantifications | 326 |
| | 5.2 Working with existential quantifications | 331 |
| | 5.2.1 Deriving existential quantifications | 331 |
| | 5.2.2 Using existential quantifications | 332 |
| | 5.3 Some examples | 338 |
| | 5.4 Methods for quantifier reasoning | 342 |
| | 5.5 Proof heuristics for first-order logic | 353 |
| | 5.5.1 Backward tactics for quantifiers | 354 |
| | 5.5.2 Forward tactics for quantifiers | 355 |
| | 5.5.3 Proof strategy for first-order logic | 371 |
| | 5.6 First-order logic semantics | 372 |
| | 5.7 Additional exercises | 386 |
| | 5.8 Chapter notes | 393 |
| 6 | Implication Chaining | 397 |
| | 6.1 Implication chains | 397 |
| | 6.2 Using sentences as justifiers | 404 |
| | 6.2.1 Nested rules | 409 |
| | 6.3 Implication chaining through sentential structure | 411 |
| | 6.4 Using chains with chain-last | 413 |
| | 6.5 Backward chains and chain-first | 415 |
| | 6.6 Equivalence chains | 417 |
| | 6.7 Mixing equational, implication, and equivalence steps | 419 |
| | 6.8 Chain nesting | 423 |
| | 6.9 Exercises | 425 |
| III | PROOFS ABOUT FUNDAMENTAL DATATYPES | 429 |
| 7 | Organizing Theory Development with Athena Modules | 431 |
| | 7.1 Introducing a module | 431 |
| | 7.2 Natural numbers using modules | 433 |
| | 7.3 Extending a module | 435 |
| | 7.4 Modules for function symbols | 436 |
| | 7.5 Additional module features | 437 |
| | 7.6 Additional module procedures | 438 |
| | 7.7 A note on indentation | 439 |

CONTENTS

ix

| | | |
|-----------|--|-----|
| 8 | Natural Number Orderings | 441 |
| | 8.1 Properties of natural number ordering functions | 441 |
| | 8.1.1 Trichotomy properties | 445 |
| | 8.1.2 Transitive and asymmetric properties | 446 |
| | 8.1.3 Less-equal properties | 448 |
| | 8.1.4 Combining ordering and arithmetic | 451 |
| | 8.2 Natural number subtraction | 453 |
| | 8.3 Ordered lists | 461 |
| | 8.4 Binary search trees | 464 |
| | 8.5 Summary and a connecting theorem | 470 |
| | 8.6 Additional exercises | 473 |
| | 8.7 Chapter notes | 474 |
| 9 | Integer Representations and Proof Mappings | 475 |
| | 9.1 Declarations and axioms | 475 |
| | 9.2 First proofs of integer properties | 477 |
| | 9.3 Another integer representation | 478 |
| | 9.4 Mappings between the signed and pair representations | 480 |
| | 9.5 Additive homomorphism property | 481 |
| | 9.6 Associativity and commutativity of integer addition | 483 |
| | 9.7 Power series | 484 |
| | 9.8 Summary and looking ahead | 487 |
| | 9.9 Additional exercises | 488 |
| 10 | Fundamental Discrete Structures | 491 |
| | 10.1 Ordered pairs | 493 |
| | 10.1.1 Representation and notation | 493 |
| | 10.1.2 Results and methods | 494 |
| | 10.2 Options | 497 |
| | 10.2.1 Representation and notation | 497 |
| | 10.2.2 Some useful results | 498 |
| | 10.3 Sets, relations, and functions | 499 |
| | 10.3.1 Representation and notation | 499 |
| | 10.3.2 Set membership, the subset relation, and set identity | 501 |
| | 10.3.3 Set operations | 508 |
| | 10.3.4 Cartesian products | 518 |
| | 10.3.5 Relations | 521 |
| | 10.3.6 Set cardinality | 529 |
| | 10.3.7 Powersets | 530 |

| | | |
|-----------|--|------------|
| 10.4 | Maps | 533 |
| 10.4.1 | Representation and notation | 533 |
| 10.4.2 | Map operations and theorems | 535 |
| 10.4.3 | Default maps | 549 |
| 10.5 | Chapter notes | 558 |
| IV | PROOFS ABOUT ALGORITHMS | 561 |
| 11 | A Binary Search Algorithm | 563 |
| 11.1 | Defining the algorithm | 563 |
| 11.1.1 | Efficiency considerations | 565 |
| 11.1.2 | Correspondence to definitions in other languages | 566 |
| 11.1.3 | Interface design | 567 |
| 11.1.4 | Testing with evaluation | 567 |
| 11.2 | First correctness properties | 569 |
| 11.3 | Specifying requirements on a function to be defined | 574 |
| 11.4 | Correctness of an optimized binary search algorithm | 575 |
| 11.5 | Summary and looking ahead | 576 |
| 11.6 | Additional exercises | 577 |
| 12 | A Fast Exponentiation Algorithm | 579 |
| 12.1 | Mathematical background | 579 |
| 12.2 | Strong induction | 581 |
| 12.3 | Properties of half | 583 |
| 12.4 | Properties of odd and even | 587 |
| 12.5 | Properties of power | 590 |
| 12.6 | Properties of fast-power | 591 |
| 12.7 | Tail recursion, a potential optimization | 593 |
| 12.8 | Transforming strong induction into ordinary induction | 596 |
| 12.9 | Measure induction | 598 |
| 12.10 | Summary and looking ahead | 599 |
| 12.11 | Additional exercises | 600 |
| 13 | Euclid's Algorithm for Greatest Common Divisors | 603 |
| 13.1 | Quotient and remainder | 603 |
| 13.2 | The division algorithm | 605 |
| 13.3 | Divisibility | 608 |
| 13.3.1 | A cancellation lemma | 610 |
| 13.3.2 | Proof of the characterization theorem | 610 |
| 13.3.3 | Additional properties of divisibility | 611 |

CONTENTS

xi

| | | |
|-----------|---|------------|
| 13.4 | Euclid's algorithm | 616 |
| 13.5 | Summary | 621 |
| 13.6 | Additional exercises | 621 |
| 13.7 | Chapter notes | 623 |
| V | PROOFS AT AN ABSTRACT LEVEL | 625 |
| 14 | Abstract Structures | 627 |
| 14.1 | Group properties | 627 |
| 14.2 | Theory refinement | 631 |
| 14.3 | Writing proofs at the level of a theory | 635 |
| 14.4 | Abstract proof method conventions | 638 |
| 14.5 | Dynamic evolution of theories | 641 |
| 14.6 | Testing abstract proofs | 642 |
| 14.7 | Group theory refinements | 644 |
| 14.7.1 | Abelian group theory | 644 |
| 14.7.2 | Multiplicative theories | 646 |
| 14.7.3 | Ring theory | 649 |
| 14.7.4 | Integral domain | 652 |
| 14.7.5 | Algebraic theory diagram | 652 |
| 14.8 | ★ Permutations as a group | 654 |
| 14.8.1 | Function theory | 654 |
| 14.8.2 | Permutation theory | 658 |
| 14.9 | Ordering properties at an abstract level | 664 |
| 14.9.1 | Binary-Relation | 664 |
| 14.9.2 | Irreflexive | 665 |
| 14.9.3 | Transitive | 666 |
| 14.9.4 | Strict partial order | 666 |
| 14.9.5 | Nonstrict partial orders | 668 |
| 14.9.6 | Strict weak order | 671 |
| 14.9.7 | A preorder | 674 |
| 14.9.8 | Strict total order | 674 |
| 14.9.9 | Lists over a strict weak order | 674 |
| 14.9.10 | Relational theory diagram | 678 |
| 14.10 | Additional exercises | 678 |
| 15 | Abstract Algorithms | 683 |
| 15.1 | An abstract binary search algorithm | 683 |
| 15.1.1 | Abstract-level binary search trees | 687 |
| 15.1.2 | Abstract-level binary search correctness theorems | 688 |

| | | |
|-----------|--|------------|
| 15.2 | An abstract fast-power algorithm | 690 |
| 15.2.1 | Raising to a power in a monoid | 690 |
| 15.2.2 | A monoid version of fast-power | 693 |
| 15.2.3 | Multiplicative version of fast power | 699 |
| 15.2.4 | A nonnumeric application | 699 |
| 16 | Algorithms on Memory Abstractions | 701 |
| 16.1 | Axioms and theorems for individual memory locations | 701 |
| 16.2 | Iterators and ranges | 706 |
| 16.2.1 | Iterator and range axioms and theorems | 708 |
| 16.2.2 | Trivial iterator: The base of a hierarchy of iterator theories | 713 |
| 16.2.3 | Forward iterators | 716 |
| 16.3 | Range count algorithm | 718 |
| 16.4 | Range replace algorithm | 721 |
| 16.5 | Range copy algorithm | 725 |
| 16.6 | Range copy-backward algorithm | 730 |
| 16.7 | Adapters: Reverse-iterator and reverse-range | 732 |
| 16.8 | Implementing copy-backward | 735 |
| 16.9 | Random access iterators | 738 |
| 16.9.1 | Relationships among iterator functions | 739 |
| 16.9.2 | New properties of the length function | 740 |
| 16.9.3 | Theorems about collecting locations | 742 |
| 16.9.4 | Ordered range | 746 |
| 16.10 | A binary search algorithm | 748 |
| 16.11 | Summary and suggestions for continued study | 753 |
| VI | PROOFS ABOUT PROGRAMMING LANGUAGES | 755 |
| 17 | A Correctness Proof for a Toy Compiler | 757 |
| 17.1 | Interpreting and compiling numeric expressions | 757 |
| 17.1.1 | Representation and notation | 757 |
| 17.1.2 | Defining the interpreter | 759 |
| 17.1.3 | An instruction set and a virtual machine | 761 |
| 17.1.4 | Compiling numeric expressions | 764 |
| 17.1.5 | Correctness | 765 |
| 17.2 | Handling errors explicitly | 773 |
| 17.2.1 | Extending the compiler with error handling | 774 |
| 17.3 | Chapter notes | 787 |

CONTENTS

xiii

| | | |
|-----------|---|-----|
| 18 | A Simple Imperative Programming Language | 789 |
| | 18.1 A simple imperative language | 789 |
| | 18.2 Semantics of expressions | 798 |
| | 18.3 Semantics of commands | 807 |
| | 18.4 * Testing the semantics | 817 |
| | 18.5 Some lemmas | 823 |
| | 18.6 Reasoning about the language | 834 |
| | 18.7 Chapter notes | 842 |
| | Appendices | 845 |
| A | Athena Reference | 847 |
| | A.1 Syntax | 847 |
| | A.2 Values | 849 |
| | A.3 Operational semantics | 851 |
| | A.4 Pattern matching | 865 |
| | A.5 Selectors | 871 |
| | A.6 Prefix syntax | 872 |
| B | Logic Programming and Prolog | 875 |
| | B.1 Basics of logic programming | 875 |
| | B.2 Examples | 878 |
| | B.3 Implementing a Prolog interpreter | 883 |
| | B.4 Integration with external Prolog systems | 888 |
| | B.5 Automating the handling of equations | 892 |
| | Bibliography | 895 |
| | Glossary | 901 |
| | Index | 909 |