Lecture Notes in Computer Science 2297

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Roland Backhouse, Roy Crole and Jeremy Gibbons
(Eds.)

Algebraic and Coalgebraic Methods in the Mathematics of Program Construction

Summer School and Workshop
Lincoln College, Oxford, UK. 10th to 14th April 2000
Lecture Notes

Springer

Preface

Program construction is about turning specifications of computer software into implementations. Doing so in a way that guarantees correctness is an undertaking requiring deep understanding of the languages and tools being used, as well as of the application domain. Recent research aimed at improving the process of program construction exploits insights from abstract algebraic tools such as lattice theory, fixpoint calculus, universal algebra, category theory and allegory theory. This book provides an introduction to these mathematical theories and how they are applied to practical problems.

The book is based on the School on *Algebraic and Co-algebraic Methods in the Mathematics of Program Construction* held in April 2000 at the University of Oxford. The School, which was sponsored under the Mathematics for Information Technology initiative of the Engineering and Physical Research Council in the UK, had the goal of introducing this research area to new PhD students in a way that would provide a sound and broad basis for their own research studies. The lecturers were chosen on the basis of a combination of a distinguished track record of research contributions in this area and an ability to present difficult material in a comprehensible way without compromising scientific rigour.

The students that attended the School had varied backgrounds to which due account was given by the lecturers in preparing their material. The experience and feedback gained during the School has been used in preparing this major revision of that material. The lecture material has also been augmented by an introductory chapter giving a detailed overview of the remaining chapters. We hope that this chapter will prove useful in allowing readers to select the chapters most relevant to their own research and to plan their further reading.

Our thanks go to all those involved in making the School a very enjoyable and successful event. Particular thanks go to the EPSRC and the London Mathematical Society for their generous financial support, to the students for their enthusiasm and positive feedback, to the lecturers for their unstinting hard work, and to Springer-Verlag for making the publication of this book possible.

Roland Backhouse
Roy Crole
Jeremy Gibbons

December 2001

Contributors

Peter Aczel: Departments of Mathematics and Computer Science, University of Manchester, Manchester M13 9PL, England. petera@cs.man.ac.uk, http://www.cs.man.ac.uk/~petera/.

Roland Backhouse: School of Computing and Information Technology, University of Nottingham, Nottingham NG8 1BB, England. rcb@cs.nott.ac.uk, http://www.cs.nott.ac.uk/~rcb/.

Richard Bird: Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, England. richard.bird@comlab.ox.ac.uk, http://www.comlab.ox.ac.uk/oucl/work/richard.bird/.

Roy Crole: Department of Mathematics and Computer Science, University of Leicester, University Road, Leicester LE1 7RH, England. roy.crole@mcs.le.ac.uk, http://www.mcs.le.ac.uk/~rcrole/.

Henk Doornbos: EverMind, Westerkade 15/4, 9718 AS Groningen, The Netherlands. henk.doornbos@evermind.com, http://www.evermind.com.

Jeremy Gibbons: Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, England. jeremy.gibbons@comlab.ox.ac.uk, http://www.comlab.ox.ac.uk/oucl/work/jeremy.gibbons/.

Bart Jacobs: Department of Computer Science, University of Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands. bart@cs.kun.nl, http://www.cs.kun.nl/~bart/.

Burghard von Karger: Christian-Albrechts-University of Kiel, Institute of Computer Science and Applied Mathematics, Olshausenstrasse 40, D-24098 Kiel, Germany. burghard.von.karger@gmx.de, http://www.informatik.uni-kiel.de/~bvk/.

Shin Cheng Mu: Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, England. shin-cheng.mu@comlab.ox.ac.uk, http://www.comlab.ox.ac.uk/oucl/work/shin-cheng.mu/.

Hilary Priestley: Oxford University Mathematical Institute, 24–29 St Giles', Oxford OX1 3LB, England. hap@maths.ox.ac.uk, http://www.stannes.ox.ac.uk/college/members/priestley.ha.html.

Table of Contents

Chapter 1. Introduction 1
Roy Crole
1 Preliminaries 1
1.1 Assumed Knowledge 1
1.2 Volume Overview 2
2 A Mathematical History Tour 3
2.1 Fixed Points 3
2.2 Induction and Coinduction 7
2.3 Types and Categories 8
2.4 Algebras and Coalgebras 9
3 Mathematics in ACMMPC 13
3.1 Chapter 2: Ordered Sets and Complete Lattices 13
3.2 Chapter 3: Introducing Algebras and Coalgebras 14
3.3 Chapter 4: Galois Connections and Fixed Point Calculus 15
3.4 Chapter 5: Calculating Functional Programs 15
3.5 Chapter 6: Algebra of Program Termination 16
3.6 Chapter 7: Exercises in Coalgebraic Specification 16
3.7 Chapter 8: Algebraic Methods for Optimization Problems 17
3.8 Chapter 9: Temporal Algebra 18

Chapter 2. Ordered Sets and Complete Lattices 20
H.A. Priestley
1 Introduction 20
2 From binary relations to diagrams 22
2.1 A fundamental example: powersets 22
2.2 Input-output relations pictorially 23
2.3 Exercise 24
2.4 Binary relations and their polars 25
2.5 Exercise 26
2.6 Summing up so far 26
3 Order, order, order, ... 26
3.1 Partial order 27
3.2 Information orderings 28
3.3 Diagrams 29
3.4 Duality: buy one, get one free 30
3.5 Bottom and top 31
3.6 Lifting 31
3.7 New posets from old: sums and products 31
3.8 Maps between posets 32
3.9 Pointwise ordering of maps 33
3.10 Up-sets: an inbred example 33

3.11 Monotone maps and up-sets . . . 34
3.12 Exercise (more on monotone maps and up-sets) . . . 35
3.13 Down is nice too . . . 35
3.14 Exercise (turning things upside down) . . . 36
3.15 The down-set operator, $\downarrow$, and the up-set operator, $\uparrow$ . . . 36
3.16 Exercise (a context explored) . . . 37
3.17 Maximal and minimal elements . . . 37
3.18 Stocktaking . . . 38
4 Lattices in general and complete lattices in particular . . . 38
4.1 Lattices . . . 38
4.2 Examples of lattices . . . 40
4.3 Distributive lattices . . . 41
4.4 Boolean algebras . . . 41
4.5 Lattices in logic . . . 42
4.6 Upper bounds and sups, lower bounds and infs . . . 42
4.7 Much ado about nothing, and about everything . . . 44
4.8 Complete lattices . . . 44
4.9 Completeness on the cheap . . . 44
4.10 A special class of complete lattices . . . 44
4.11 Suprema, infima, and monotone maps . . . 45
5 Complete lattices, concretely: closure systems and closure operators . . . 46
5.1 A useful technical remark . . . 46
5.2 Complete lattices of sets . . . 46
5.3 Closure systems . . . 46
5.4 Closure systems . . . 47
5.5 Examples . . . 48
5.6 From a complete lattice to a closure system . . . 48
5.7 Defining closure operators . . . 48
5.8 New complete lattices from old: from a closure operator to a complete lattice . . . 49
5.9 Closure operators more concretely . . . 50
6 Galois connections: basics . . . 50
6.1 Introduction . . . 50
6.2 Lattice representation via Galois connections . . . 51
6.3 Galois connections from binary relations: method I . . . 52
6.4 Galois connections and algebras—a fleeting glimpse . . . 53
6.5 Galois connections by sectioning . . . 53
6.6 Galois connections from binary relations: method II . . . 54
6.7 Galois connections: basic properties . . . 54
6.8 $\triangleright$ and $\triangleleft$ have isomorphic images . . . 55
6.9 Equivalent definitions for Galois connections . . . 56
6.10 The good (and less good) behaviour of Galois maps . . . 57
6.11 Uniqueness of adjoints: $\triangleright$ from $\triangleleft$ and $\triangleleft$ from $\triangleright$ . . . 58
6.12 Exercise (surjective and injective Galois maps) . . . 59
6.13 A look ahead . . . 59

6.14 Existence of adjoints: a technical lemma . . . 59
6.15 Existence theorem for adjoints . . . 60
6.16 Postscript . . . 61
7 Making connections, conceptually . . . 61
7.1 From a Galois connection to a closure operator . . . 61
7.2 From a closure operator to a Galois connection . . . 62
7.3 Contexts and concepts: re-setting the scene . . . 62
7.4 Ordering concepts . . . 62
7.5 Three for the price of one: a trinity of complete lattices . . . 63
7.6 Manufacturing concepts . . . 63
7.7 Density: generating all concepts via γ or μ . . . 64
7.8 From a complete lattice to a concept lattice . . . 65
7.9 The case for the defence . . . 65
7.10 Summing up . . . 66
8 The existence of fixed points . . . 66
8.1 Fixed points and least fixed points: definitions . . . 67
8.2 Characterizing least fixed points via least prefix points . . . 67
8.3 The Knaster–Tarski Fixed Point Theorem . . . 67
8.4 Exercise . . . 67
8.5 Exercise . . . 68
8.6 From complete lattices to CPOs . . . 68
8.7 A sense of direction . . . 68
8.8 Exercise (sups and directed sups related) . . . 69
8.9 CPOs . . . 69
8.10 Directed sets and continuity . . . 70
8.11 New CPOs from old . . . 70
8.12 Fixed Point Theorem for a continuous function on a CPO . . . 71
8.13 From continuity to monotonicity . . . 71
8.14 An assumption: Zorn's Lemma (ZL), CPO form . . . 71
8.15 The Fixed Point Theorem for a monotone endofunction on a CPO 72
8.16 Exercise . . . 72
8.17 Exercise . . . 72
8.18 Concluding remarks . . . 73
9 Speaking categorically . . . 73
9.1 Categories . . . 73

Chapter 3. Introducing Algebras and Coalgebras . . . 78
Peter Aczel
1 Introduction . . . 78
2 Terms . . . 79
2.1 Variable-free Terms . . . 79
2.2 Terms as Set Theoretical Objects . . . 80
2.3 Terms with Variables . . . 81
2.4 Initial (F, X)-Algebras . . . 82
2.5 Substitution . . . 82
3 Trees . . . 83

3.1 Variable-free Trees . . . 83
3.2 Representing Terms as Well-founded Trees . . . 83
3.3 Corecursion . . . 83
3.4 Set Theoretical Representation of Trees . . . 84
3.5 Trees with Variables . . . 85
3.6 Substitution . . . 86
3.7 Solution Property . . . 86
4 The Monad of a Substitution System . . . 87

Chapter 4. Galois Connections and Fixed Point Calculus . . . 88
Roland Backhouse
1 Introduction . . . 88
1.1 Fixed Point Equations . . . 88
1.2 Languages . . . 88
1.3 Functions . . . 89
1.4 Datatypes . . . 90
1.5 Galois Connections . . . 90
1.6 Basic Assumptions . . . 91
1.7 Issues and Applications . . . 91
2 Galois Connections — Introductory Examples . . . 92
2.1 Simple Examples . . . 92
2.2 The Floor Function . . . 95
3 Identifying Galois Connections . . . 99
3.1 Symmetric Definitions . . . 99
3.2 Universal Property . . . 101
3.3 Commutativity Properties . . . 102
4 Pair Algebras . . . 104
4.1 Infima and Suprema . . . 106
4.2 Extremum Preservation Properties . . . 108
4.3 Existence Theorem . . . 110
5 Fixed Points . . . 114
5.1 Prefix Points . . . 114
5.2 A First Example . . . 118
5.3 Kleene Algebra . . . 121
6 Fixed Point Calculus . . . 126
6.1 Basic Rules . . . 127
6.2 Fusion . . . 128
6.3 Uniqueness . . . 133
6.4 Parameterised Prefix Points . . . 135
6.5 Mutual Recursion . . . 140
6.6 An Illustration — Arithmetic Expressions . . . 142
7 Further Reading . . . 145

Chapter 5. Calculating Functional Programs 148
Jeremy Gibbons
1 Introduction 148
1.1 Why calculate programs? 148
1.2 Functional programming 149
1.3 Universal properties 149
1.4 The categorical approach to datatypes 152
1.5 The pair calculus 155
1.6 Bibliographic notes 158
1.7 Exercises 158
2 Recursive datatypes in the category $\mathcal{S}et$ 159
2.1 Overview 159
2.2 Monomorphic datatypes 160
2.3 Folds 161
2.4 Polymorphic datatypes 163
2.5 Properties of folds 164
2.6 Co-datatypes and unfolds 167
2.7 . . . and never the twain shall meet 169
2.8 Bibliographic notes 170
2.9 Exercises 170
3 Recursive datatypes in the category $\mathcal{C}po$ 174
3.1 The category $\mathcal{C}po$ 174
3.2 Continuous algebras 176
3.3 The pair calculus again 177
3.4 Hylomorphisms 178
3.5 Bibliographic notes 181
3.6 Exercises 181
4 Applications 183
4.1 A simple compiler 184
4.2 Monads and comonads 187
4.3 Breadth-first traversal 192
4.4 Bibliographic notes 194
4.5 Exercises 195
5 Bibliography 196
6 Appendix: Implementation in Haskell 198
6.1 Products 198
6.2 Sums 199
6.3 Functors 199
6.4 Datatypes 200
6.5 Folds and unfolds 200
6.6 Lists 200
6.7 Trees 201
6.8 Quicksort 202

Chapter 6. Algebra of Program Termination 204
Henk Doornbos and Roland Backhouse
1 Introduction 204
2 Imperative Programming and Well-founded Relations 204
2.1 Relation Algebra 205
2.2 Imperative Programming 207
2.3 Domains and Division 209
2.4 Well-Foundedness Defined 211
2.5 Totality of **while** statements 215
2.6 Induction Principles 216
2.7 Admits-induction Implies Well-Founded 218
3 Hylo Equations 219
3.1 Relators and Hylos 220
3.2 Hylo Programs 221
3.3 Intermediate data structures 228
3.4 The Hylo Theorem 230
3.5 Reducing problem size 231
3.6 A calculus of F-reductivity 233

Chapter 7. Exercises in Coalgebraic Specification 237
Bart Jacobs
1 Introduction 237
2 Mathematical preliminaries 239
3 Specification of groups and vector spaces 241
4 A first coalgebraic specification: binary trees 244
4.1 Elements of binary trees 247
5 Bisimulations and bisimilarity 248
6 Invariants 253
7 Temporal logic for coalgebras 255
7.1 A concrete description of $\Box$ and $\Diamond$ for binary trees 256
7.2 Using $\Box$ and $\Diamond$ for specification and verification of binary trees . . . 259
8 Towards a μ-calculus for coalgebras 261
9 A case study: Peterson's mutual exclusion algorithm 265
9.1 Peterson's solution for mutual exclusion 265
9.2 Dealing with time in coalgebraic specification 267
9.3 Class-valued methods 269
9.4 Peterson's algorithm in coalgebraic specification 270
10 Refinements between coalgebraic specifications 275
11 Conclusion 277

Chapter 8. Algebraic Methods for Optimization Problems 281
Richard Bird, Jeremy Gibbons and Shin-Cheng Mu
1 Introduction 281
1.1 Bibliographic notes 283
2 The algebra of relations 283
2.1 Relations 283

2.2 Special kinds of relation . . . 286
2.3 Breadth . . . 287
2.4 Folds . . . 287
2.5 Bibliographic notes . . . 290
2.6 Exercises . . . 290
3 Optimization problems . . . 291
3.1 The Eilenberg-Wright Lemma . . . 292
3.2 Preorders . . . 293
3.3 Monotonicity . . . 293
3.4 Minimum . . . 293
3.5 The Greedy Theorem . . . 294
3.6 Thinning . . . 295
3.7 Bibliographic notes . . . 297
3.8 Exercises . . . 297
4 Optimal bracketing . . . 298
4.1 Representation . . . 299
4.2 The Converse-of-a-Function Theorem . . . 300
4.3 Spines . . . 300
4.4 An application of thinning . . . 301
4.5 An application of greediness . . . 302
4.6 Refinement of the greedy algorithm to a program . . . 303
4.7 Summary . . . 304
4.8 The Haskell program . . . 304
4.9 Bibliographic notes . . . 304
4.10 Exercises . . . 304
5 Bibliography . . . 306

Chapter 9. Temporal Algebra . . . 308
Burghard von Karger
1 Introduction . . . 308
2 Preliminaries . . . 310
2.1 Lattices . . . 310
2.2 Galois Connections . . . 311
2.3 Boolean Algebra . . . 312
2.4 Fixed Points . . . 317
2.5 Regular Algebra . . . 319
2.6 Iteration . . . 319
2.7 Repetition . . . 321
3 Galois Algebra . . . 321
3.1 Definition and Basic Properties . . . 322
3.2 New Algebras From Old . . . 325
3.3 Diamonds and Boxes . . . 326
3.4 'Until' and 'Since' . . . 329
3.5 Confluence . . . 331
3.6 Linearity . . . 334
3.7 Infinity . . . 335

4 Sequential Algebra 336
4.1 Observations 337
4.2 Lifting to Sets 342
4.3 Sequential Set Algebras 345
4.4 Abstract Sequential Algebras 351
4.5 Time-wise duality 353
5 Relational Laws of Sequential Algebra 354
5.1 Basic Laws 355
5.2 Predicates 358
5.3 Left and Right Domain 360
6 Interval Calculi 364
6.1 Somewhere and Everywhere 364
6.2 Importability 368
6.3 Engineer's Induction 370
6.4 Finite and Infinite Observations 372
6.5 Measuring Time 375
6.6 Phase Calculus 376
6.7 Duration Calculus 380
7 Conclusion 382