HANDBOOK OF LOGIC IN ARTIFICIAL INTELLIGENCE AND LOGIC PROGRAMMING

Volume 5
Logic Programming

Edited by
DOV M. GABBAY
and
C. J. HOGGER
Imperial College of Science, Technology and Medicine
London
and
J. A. ROBINSON
Syracuse University, New York

CLARENDON PRESS • OXFORD
1998

Contents

List of contributors xiv

Introduction: Logic and Logic Programming Languages
Michael J. O'Donnell 1
1 Introduction 1
 1.1 Motivation 1
 1.2 A notational apology 3
2 Specifying logic programming languages 7
 2.1 Semantic systems and semantic consequences . . . 7
 2.2 Query Systems, questions and answers 11
 2.3 Examples of logic programming languages 15
3 Implementing logic programming languages 37
 3.1 Proof systems 37
 3.2 Soundness and completeness of proof systems . . . 40
 3.3 Programming systems 44
 3.4 Soundness and completeness of programming systems 49
 3.5 Proof-theoretic foundations for logic programming 56
4 The uses of semantics 57
 4.1 Logical semantics vs. denotational semantics 57
 4.2 Logical semantics vs. initial/final-algebra and Herbrand semantics 58

Equational Logic Programming
Michael J. O'Donnell 69
1 Introduction to equational logic programming 69
 1.1 Survey of prerequisites 69
 1.2 Motivation for programming with equations 71
 1.3 Outline of the chapter 74
2 Proof systems for equational logic 75
 2.1 Inferential proofs 75
 2.2 Term rewriting proofs 78
 2.3 The confluence property and the completeness of term rewriting 81
3 Term rewriting proof strategies 96

3.1 Complete and outer most complete rewriting sequences 97
3.2 Sequentiality analysis and optimal rewriting 100
4 Algorithms and data structures to implement equational languages 111
4.1 Data structures to represent terms 111
4.2 Pattern-matching and sequencing methods 120
4.3 Driving procedures for term rewriting 129
5 Compiling efficient code from equations 137
6 Parallel implementation 139
7 Extensions to equational logic programming 141
7.1 Incremental infinite input and output 141
7.2 Solving equations 147
7.3 Indeterminate evaluation in subset logic 149
7.4 Relational rewriting 151

Proof Procedures for Logic Programming
Donald W. Loveland and Gopalan Nadathur 163
1 Building the framework: the resolution procedure 163
1.1 The resolution procedure 164
1.2 Linear resolution refinements 175
2 The logic programming paradigm 186
2.1 Horn clause logic programming 186
2.2 A framework for logic programming 190
2.3 Abstract logic programming languages 198
3 Extending the logic programming paradigm 212
3.1 A language for hypothetical reasoning 213
3.2 Near-Horn Prolog 219
4 Conclusion 229

The Role of Abduction in Logic Programming
A. C. Kakas, R. A. Kowalski and F. Toni 235
1 Introduction 236
1.1 Abduction in logic 237
1.2 Integrity constraints 241
1.3 Applications 243
2 Knowledge assimilation 244
3 Default reasoning viewed as abduction 249
4 Negation as failure as abduction 254
4.1 Logic programs as abductive frameworks 255
4.2 An abductive proof procedure for LP 257
4.3 An argumentation-theoretic interpretation 263

4.4 An argumentation-theoretic interpretation of the abductive proof procedure 267
5 Abductive logic programming 269
5.1 Generalized stable model semantics 270
5.2 An abductive proof procedure for ALP 273
5.3 An argumentation-theoretic interpretation of the abductive proof procedure for ALP 277
5.4 Computation of abduction through TMS 279
5.5 Simulation of abduction 279
5.6 Abduction through deduction from the completion 285
5.7 Abduction and constraint logic programming . . . 286
6 Extended logic programming 288
6.1 Answer set semantics 289
6.2 Restoring consistency of answer sets 290
6.3 Rules and exceptions in LP 293
6.4 (Extended) Logic Programming without Negation as Failure 295
6.5 An argumentation-theoretic approach to ELP . . . 297
6.6 A methodology for default reasoning with explicit negation 299
6.7 ELP with abduction 300
7 An abstract argumentation-based framework for default reasoning 300
8 Abduction and truth maintenance 303
8.1 Justification-based truth maintenance 304
8.2 Assumption-based truth maintenance 305
9 Conclusions and future work 307

Semantics for Disjunctive and Normal Disjunctive Logic Programs
Jorge Lobo, Jack Minker and Arcot Rajasekar 325
1 Introduction 325
2 Positive consequences in logic programs 327
2.1 Definite logic programming 328
2.2 Disjunctive logic programming 330
3 Negation in logic programs 337
3.1 Negation in definite logic programs 337
3.2 Negation in disjunctive logic programs 338
4 Normal or general disjunctive logic programs 340
4.1 Stratified definite logic programs 341
4.2 Stratified disjunctive logic programs 343
4.3 Well-founded and generalized well-founded logic programs 346

4.4 Generalized disjunctive well-founded semantics . . 346
5 Summary 347
6 Addendum 349

Negation as Failure, Completion and Stratification
J. C. Shepherdson 356
1 Overview/introduction 356
1.1 Negation as failure, the closed world assumption and the Clark completion 356
1.2 Incompleteness of NF for *comp*(*P*) 359
1.3 Floundering, an irremovable source of incompleteness 359
1.4 Cases where SLDNF-resolution is complete for *comp*(*P*) 361
1.5 Semantics for negation via special classes of model 362
1.6 Semantics for negation using non-classical logics . . 363
1.7 Constructive negation: an extension of negation as failure 364
1.8 Concluding remarks 365
2 Main body 365
2.1 Negation in logic programming 365
2.2 Negation as failure; SLDNF-resolution 367
2.3 The closed world assumption, CWA(P) 370
2.4 The Clark completion, *comp*(*P*) 374
2.5 Definite Horn clause programs 384
2.6 Three-valued logic 385
2.7 Cases where SLDNF-resolution is complete for *comp*(*P*): hierarchical, stratified and call-consistent programs. 391
2.8 Semantics for negation in terms of special classes of models 393
2.9 Constructive negation; an extension of negation as failure 402
2.10 Modal and autoepistemic logic 406
2.11 Deductive calculi for negation as failure 409

Meta-Programming in Logic Programming
P. M. Hill and J. Gallagher 421
1 Introduction 422
1.1 Theoretical foundations 423
1.2 Applications 425
1.3 Efficiency improvements 426
1.4 Preliminaries 427

2 The non-ground representation 429
2.1 The representation 431
2.2 Reflective predicates 434
2.3 Meta-programming in Prolog 439
3 The ground representation 440
3.1 The representation 442
3.2 Reflective predicates 448
3.3 The language Gödel and meta-programming 453
4 Self-applicability 459
4.1 Separated meta-programming 460
4.2 Amalgamated meta-programming 461
4.3 Ambivalent logic 467
5 Dynamic meta-programming 468
5.1 Constructing programs 468
5.2 Updating programs 471
5.3 The three wise men problem 473
5.4 Transforming and specializing programs 478
6 Specialization of meta-programs 481
6.1 Logic program specialization 481
6.2 Specialization and compilation 487
6.3 Self-applicable program specializers 488
6.4 Applications of meta-program specialization 489

Higher-Order Logic Programming
Gopalan Nadathur and Dale Miller 499
1 Introduction 500
2 A motivation for higher-order features 502
3 A higher-order logic 510
3.1 The language 510
3.2 Equality between terms 513
3.3 The notion of derivation 517
3.4 A notion of models 519
3.5 Predicate variables and the subformula property . 522
4 Higher-order Horn clauses 523
5 The meaning of computations 528
5.1 Restriction to positive terms 529
5.2 Provability and operational semantics 534
6 Towards a practical realization 537
6.1 The higher-order unification problem 538
6.2 $\mathcal{P}$ derivations 541
6.3 Designing an actual interpreter 546
7 Examples of higher-order programming 549
7.1 A concrete syntax for programs 549

7.2 Some simple higher-order programs 552
7.3 Implementing tactics and tacticals 556
7.4 A comparison with functional programming 560
8 Using λ-terms as data structures 561
8.1 Implementing an interpreter for Horn clauses . . . 563
8.2 Dealing with functional programs as data 565
8.3 A limitation of higher-order Horn clauses 572
9 Hereditary Harrop formulas 574
9.1 Universal quantifiers and implications in goals . . . 574
9.2 Recursion over structures with binding 577
10 Conclusion 584

Constraint Logic Programming: A Survey
Joxan Jaffar and Michael J. Maher 591
1 Introduction 592
1.1 Constraint languages 593
1.2 Logic Programming 595
1.3 CLP languages 596
1.4 Synopsis 598
1.5 Notation and terminology 599
2 Constraint domains 601
3 Logical semantics 608
4 Fixedpoint semantics 609
5 Top-down execution 611
6 Soundness and completeness results 615
7 Bottom-up execution 617
8 Concurrent constraint logic programming 619
9 Linguistic extensions 621
9.1 Shrinking the computation tree 621
9.2 Complex constraints 623
9.3 User-defined constraints 624
9.4 Negation 625
9.5 Preferred solutions 626
10 Algorithms for constraint solving 628
10.1 Incrementality 628
10.2 Satisfiability (non-incremental) 630
10.3 Satisfiability (incremental) 633
10.4 Entailment 637
10.5 Projection 640
10.6 Backtracking 643
11 Inference engine 645
11.1 Delaying/wakeup of goals and constraints 645
11.2 Abstract machine 651

11.3 Parallel implementations 657
12 Modelling of complex problems 658
12.1 Analysis and synthesis of analog circuits 658
12.2 Options trading analysis 660
12.3 Temporal reasoning 664
13 Combinatorial search problems 665
13.1 Cutting stock 666
13.2 DNA sequencing 668
13.3 Scheduling 670
13.4 Chemical hypothetical reasoning 671
13.5 Propositional solver 674
14 Further applications 675

Transformation of Logic Programs
Alberto Pettorossi and Maurizio Proietti 697
1 Introduction 697
2 A preliminary example 701
3 Transformation rules for logic programs 704
3.1 Syntax of logic programs 704
3.2 Semantics of logic programs 706
3.3 Unfold/fold rules 707
4 Correctness of the transformation rules 715
4.1 Reversible transformations 716
4.2 A derived goal replacement rule 719
4.3 The unfold/fold proof method 721
4.4 Correctness results for definite programs 723
4.5 Correctness results for normal programs 736
5 Strategies for transforming logic programs 742
5.1 Basic strategies 745
5.2 Techniques which use basic strategies 747
5.3 Overview of other techniques 760
6 Partial evaluation and program specialization 764
7 Related methodologies for program development 771

Index 789