Programming with higher-order logic

© Dale Miller and Gopalan Nadathur
Draft: 26 October 2011, 07:23:22   Revision 5420M

Dale Miller
INRIA Saclay - Île de France
LIX / École Polytechnique
91128 Palaiseau Cedex FRANCE

Gopalan Nadathur
Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455 USA
## Contents

**List of Figures**

| List of Figures | 6 |

**Preface**

| Preface | 11 |

**Introduction**

| Connections between logic and computation | 13 |
| Logical primitives and programming expressivity | 14 |
| The meaning of “higher-order logic” | 15 |
| Presentation style | 17 |
| Prerequisites | 17 |
| Organization of the book | 18 |

**1 First-Order Terms and Representations of Data**

| Sorts and type constructors | 21 |
| Type expressions | 22 |
| Typed first-order terms | 24 |
| Representing symbolic objects | 28 |
| Representing binary trees | 28 |
| Representing logical formulas | 30 |
| Representing imperative programs | 31 |
| Unification of typed first-order terms | 32 |
| Bibliographic notes | 37 |

**2 First-Order Horn Clauses**

| First-order formulas | 39 |
| Logic programming and search semantics | 42 |
| Horn clauses and their computational interpretation | 46 |
| Programming with first-order Horn clauses | 48 |
| Concrete syntax for program clauses | 48 |
| Interacting with the λProlog system | 50 |
| Reachability in a finite state machine | 53 |
| Defining relations over recursively structured data | 55 |
| Programming over abstract syntax representations | 56 |
| Pragmatic aspects of computing with Horn clauses | 58 |
| The relationship to logical notions | 60 |
| The cut rule and cut-elimination | 60 |
| Different presentations of $\forall \text{OHC}$ | 61 |
| The meaning and use of types | 63 |
| Types and the categorization of expressions | 63 |
| Polymorphic typing | 63 |
3 First-Order Hereditary Harrop Formulas

1 The syntax of goals and program clauses
2 Implicational goals
2.1 Inferences among propositional clauses
2.2 Hypothetical reasoning
3 Universally quantified goals
3.1 Substitution and quantification
3.2 Quantification can link goals and clauses
4 The relationship to logical notions
4.1 Classical vs intuitionistic logic
4.2 Intuitionistic vs minimal logic
4.3 Notable subsets of fohh
5 Bibliographic notes

4 Typed Lambda Terms and Formulas

1 Syntax for λ-terms and formulas
2 The rules for λ-conversion
3 Some properties of λ-conversion
4 Unification problems as quantified equalities
4.1 Simplifying quantifier prefixes
4.2 Unifiers, solutions, and empty types
4.3 Examples of unification problems and their solutions
5 Solving unification problems
6 Some hard unification problems
6.1 Solving Post correspondence problems
6.2 Solving Diophantine equations
7 Bibliographic notes

5 Using Quantification at Higher-Order Types

1 Atomic formulas in higher-order logic programs
1.1 Flexible atoms as heads of clauses
1.2 Logical symbols within atomic formulas
2 Higher-order logic programming languages
2.1 Higher-order Horn clauses
2.2 Higher-order hereditary Harrop formulas
2.3 Extended higher-order hereditary Harrop formulas
3 Examples of higher-order programming
4 Flexible atoms as goals
5 Reasoning about higher-order programs
6 Defining some of the logical constants
7 The conditional and negation-as-failure
8 Using λ-terms as functions
8.1 Some basic computations with functional expressions
8.2 Functional difference lists
9 Higher-order unification is not a panacea
10 Comparison with functional programming
11 Bibliographic notes
## 6 Mechanisms for Structuring Large Programs

1. Desiderata for modular programming ........................................... 127
2. A modules language ............................................................... 128
3. Matching signatures and modules .............................................. 132
4. The logical interpretation of modules ....................................... 135
   4.1 Existential quantification in program clauses ......................... 135
   4.2 A module as a logical formula ........................................... 137
   4.3 Interpreting queries against modules .................................. 137
   4.4 Module accumulation as scoped inlining of code .................... 138
5. Some programming aspects of the modules language ..................... 139
   5.1 Hiding and abstract datatypes .......................................... 139
   5.2 Code extensibility and modular composition ............................ 141
   5.3 Signature accumulation and parametrization of modules ............. 141
   5.4 Higher-order programming and predicate visibility ................... 143
6. Implementation considerations ............................................... 145
7. Bibliographic notes ............................................................. 145

## 7 Computations over $\lambda$-Terms

1. Representing objects with binding structure ................................ 147
   1.1 Encoding logical formulas with quantifiers ........................... 148
   1.2 Encoding untyped $\lambda$-terms ........................................ 149
   1.3 Properties of the encoding of binding .................................. 150
2. Realizing object-level substitution ......................................... 151
3. Mobility of binders .................................................................. 153
4. Computing with untyped $\lambda$-terms ...................................... 155
   4.1 Computing normal forms .................................................. 155
   4.2 Reduction based on paths through terms ................................ 156
   4.3 Type inference .................................................................. 158
   4.4 Translating to and from de Bruijn syntax ............................... 160
5. Computations over first-order formulas .................................... 162
6. Specifying object-level substitution ......................................... 165
7. The $\lambda$-tree approach to abstract syntax ................................ 169
8. The $L_\lambda$ subset of $\lambda$Prolog ........................................... 169
9. Bibliographic notes ............................................................... 172

## 8 Unification of $\lambda$-Terms

1. Properties of the higher-order unification problem ....................... 175
2. A procedure for checking for unifiability ................................... 177
   2.1 Simplification of rigid-rigid equations ................................. 177
   2.2 Substitutions for equations between flexible and rigid terms ....... 177
   2.3 The iterative transformation of unification problems ................. 178
   2.4 Unification problems with only flexible-flexible equations ......... 179
   2.5 Non-termination of reductions .......................................... 180
   2.6 Matching trees .................................................................. 180
3. Higher-order pattern unification .............................................. 182
4. Pragmatic aspects of higher-order unification .............................. 185
5. Bibliographic notes ............................................................. 186
9 Implementing Proof Systems 189
  1 Deduction in propositional intuitionistic logic .......................... 189
  2 Encoding natural deduction for intuitionistic logic ..................... 191
  3 A theorem-prover for classical logic .................................... 194
  4 A general architecture for theorem provers .............................. 198
    4.1 Goals and tactics ................................................. 198
    4.2 Combining tactics into proof strategies .......................... 200
  5 Bibliographic notes .................................................. 202

10 Computations over Functional Programs 205
  1 The miniFP programming language ..................................... 205
  2 Specifying evaluation for miniFP programs ............................ 208
    2.1 A big-step style specification .................................. 208
    2.2 A specification using evaluation contexts ....................... 210
  3 Manipulating functional programs ..................................... 211
    3.1 Partial evaluation of miniFP programs ......................... 211
    3.2 Transformation to continuation passing style .................. 212
  4 Bibliographic notes .................................................. 213

11 Encoding a Process Calculus Language 217
  1 Representing the expressions of the $\pi$-calculus .................. 217
  2 Specifying one-step transitions ...................................... 219
  3 Animating $\pi$-calculus expressions .................................. 221
  4 May versus must judgments ........................................... 223
  5 Mapping the $\lambda$-calculus into the $\pi$-calculus ................. 226
  6 Bibliographic notes .................................................. 228

A The Teyjus System 229
  1 An overview of the Teyjus system .................................... 229
  2 Interacting with the Teyjus system .................................. 230
  3 Using modules within the Teyjus system ................................ 233
  4 Special features of the Teyjus system ................................ 235
    4.1 Built-In types and predicates .................................. 236
    4.2 Deviations from the language assumed in this book ............. 237

Bibliography 238

Index 249