



Calhoun: The NPS Institutional Archive
DSpace Repository

Faculty and Researchers

Faculty and Researchers' Publications

1988

Artificial Intelligence through Prolog by Neil C. Rowe

Rowe, Neil C.

Prentice-Hall

<http://hdl.handle.net/10945/36984>

Downloaded from NPS Archive: Calhoun



Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

Table of contents

Preface

Acknowledgements

To the reader

1. Introduction

- 1.1 What artificial intelligence is about
- 1.2 Understanding artificial intelligence
- 1.3 Preview

2. Representing facts

- 2.1 Predicates and predicate expressions
- 2.2 Predicates indicating types
- 2.3 About types
- 2.4 Good naming
- 2.5 Property predicates
- 2.6 Predicates for relationships
- 2.7 Semantic networks
- 2.8 Getting facts from English descriptions
- 2.9 Predicates with three or more arguments
- 2.10 Probabilities
- 2.11 How many facts do we need?

3. Variables and queries

- 3.1 Querying the facts
- 3.2 Queries with one variable
- 3.3 Multi-directional queries
- 3.4 Matching alternatives
- 3.5 Multi-condition queries
- 3.6 Negative predicate expressions
- 3.7 Some query examples
- 3.8 Loading a database
- 3.9 Backtracking
- 3.10 A harder backtracking example: superbosses
- 3.11 Backtracking with "not"s
- 3.12 The generate-and-test scheme
- 3.13 Backtracking with "or"s (*)

- 3.14 Implementation of backtracking
- 3.15 About long examples

4. Definitions and inferences

- 4.1 Rules for definitions
- 4.2 Rule and fact order
- 4.3 Rules as programs
- 4.4 Rules in natural language
- 4.5 Rules without right sides
- 4.6 Postponed binding
- 4.7 Backtracking with rules
- 4.8 Transitivity inferences
- 4.9 Inheritance inferences
- 4.10 Some implementation problems for transitivity and inheritance
- 4.11 A longer example: some traffic laws
- 4.12 Running the traffic lights program
- 4.13 Declarative programming

5. Arithmetic and lists in Prolog

- 5.1 Arithmetic comparisons
- 5.2 Arithmetic assignment
- 5.3 Reversing the "is"
- 5.4 Lists in Prolog
- 5.5 Defining some list-processing predicates
- 5.6 List-creating predicates
- 5.7 Combining list predicates
- 5.8 Redundancy in definitions
- 5.9 An example: dejargonizing bureaucratese (*)

6. Control structures for rule-based systems

- 6.1 Backward-chaining control structures
- 6.2 Forward chaining
- 6.3 A forward chaining example
- 6.4 Hybrid control structures
- 6.5 Order variants
- 6.6 Partitioned control structures
- 6.7 Meta-rules
- 6.8 Decision lattices
- 6.9 Concurrency in control structures
- 6.10 And-or-not lattices
- 6.11 Randomness in control structures
- 6.12 Grammars for interpreting languages (*)

7. Implementation of rule-based systems

- 7.1 Implementing backward chaining
- 7.2 Implementing virtual facts in caching
- 7.3 Input coding
- 7.4 Output coding
- 7.5 Intermediate predicates
- 7.6 An example program
- 7.7 Running the example program
- 7.8 Partitioned rule-based systems
- 7.9 Implementing the rule-cycle hybrid
- 7.10 Implementing pure forward chaining (*)
- 7.11 Forward chaining with "not"s (*)
- 7.12 General iteration with "forall" and "doall" (*)
- 7.13 Input and output of forward chaining (*)
- 7.14 Rule form conversions (*)
- 7.15 Indexing of predicates (*)
- 7.16 Implementing meta-rules (*)
- 7.17 Implementing concurrency (*)
- 7.18 Decision lattices: a compilation of a rule-based system (*)
- 7.19 Summary of the code described in the chapter (*)

8. Representing uncertainty in rule-based systems

- 8.1 Probabilities in rules
- 8.2 Some rules with probabilities
- 8.3 Combining evidence assuming statistical independence
- 8.4 Prolog implementation of independence-assumption "and-combination"
- 8.5 Prolog implementation of independence-assumption "or-combination"
- 8.6 The conservative approach
- 8.7 The liberal approach and others
- 8.8 Negation and probabilities
- 8.9 An example: fixing televisions
- 8.10 Graphical representation of probabilities in rule-based systems
- 8.11 Getting probabilities from statistics
- 8.12 Probabilities derived from others
- 8.13 Subjective probabilities
- 8.14 Maximum-entropy probabilities (*)
- 8.15 Consistency (*)

9. Search

- 9.1 Changing worlds
- 9.2 States
- 9.3 Three examples
- 9.4 Operators

- 9.5 Search as graph traversal
- 9.6 The simplest search strategies: depth-first and breadth-first
- 9.7 Heuristics
- 9.8 Evaluation functions
- 9.9 Cost functions
- 9.10 Optimal-path search
- 9.11 A route-finding example
- 9.12 Special cases of search
- 9.13 How hard is a search problem?
- 9.14 Backward chaining versus forward chaining (*)
- 9.15 Using probabilities in search (*)
- 9.16 Another example: visual edge-finding as search (*)

10. Implementing search

- 10.1 Defining a simple search problem
- 10.2 Defining a search problem with fact-list states
- 10.3 Implementing depth-first search
- 10.4 A depth-first example
- 10.5 Implementing breadth-first search
- 10.6 Collecting all items that satisfy a predicate expression
- 10.7 The cut predicate
- 10.8 Iteration with the cut predicate (*)
- 10.9 Implementing best-first search (*)
- 10.10 Implementing A* search (*)
- 10.11 Implementing search with heuristics (*)
- 10.12 Compilation of search (*)

11. Abstraction in search

- 11.1 Means-ends analysis
- 11.2 A simple example
- 11.3 Partial state description
- 11.4 Implementation of means-ends analysis
- 11.5 A harder example: flashlight repair
- 11.6 Running the flashlight program
- 11.7 Means-ends versus other search methods
- 11.8 Modeling real-world uncertainty (*)
- 11.9 Procedural nets (*)

12. Abstraction of facts

- 12.1 Partitioning facts
- 12.2 Frames and slots
- 12.3 Slots qualifying other slots
- 12.4 Frames with components

- 12.5 Frames as forms: memos
- 12.6 Slot inheritance
- 12.7 Part-kind inheritance
- 12.8 Extensions versus intensions
- 12.9 Procedural attachment
- 12.10 Frames in Prolog
- 12.11 Example of a frame lattice
- 12.12 Expectations from slots
- 12.13 Frames for natural language understanding (*)
- 12.14 Multiple inheritance (*)
- 12.15 A multiple inheritance example: custom operating systems (*)

13. Problems with many constraints

- 13.1 Two examples
- 13.2 Rearranging long queries without local variables
- 13.3 Some mathematics
- 13.4 Rearranging queries with local variables
- 13.5 Rearranging queries based on dependencies
- 13.6 Summary of guidelines for optimal query arrangements
- 13.7 Rearrangement and improvement of the photo interpretation query
- 13.8 Dependency-based backtracking
- 13.9 Reasoning about possibilities
- 13.10 Using relaxation for the photo interpretation example
- 13.11 Quantifying the effect (*)
- 13.12 Formalization of pure relaxation
- 13.13 Another relaxation example: cryptarithmic
- 13.14 Implementation of pure relaxation (*)
- 13.15 Running a cryptarithmic relaxation (*)
- 13.16 Implementing double relaxation (*)

14. A more general logic programming

- 14.1 Logical limitations of Prolog
- 14.2 The logical (declarative) meaning of Prolog rules and facts
- 14.3 Extending Prolog rules
- 14.4 More about clause form
- 14.5 Resolution
- 14.6 Resolution with variables
- 14.7 Three important applications of resolution
- 14.8 Resolution search strategies
- 14.9 Implementing resolution without variables (*)

15. Testing and debugging of artificial intelligence programs

- 15.1 The gold standard

- 15.2 Cases
- 15.3 Focusing on bugs
- 15.4 Exploiting pairs of similar cases
- 15.5 Composite results
- 15.6 Numbers in comparisons
- 15.7 Preventive measures
- 15.8 Supporting intuitive debugging
- 15.9 Evaluating cooperativeness
- 15.10 On problems unsuitable for artificial intelligence

Appendix A: basics of logic

Appendix B: Basics of recursion

Appendix C: Basics of data structures

Appendix D: summary of the Prolog dialect used in this book

- D.1 Managing facts and rules
- D.2 The format of facts, rules and queries
- D.3. Program layout
- D.4. Lists
- D.5. Numbers
- D.6. Output and input
- D.7. Strings
- D.8. Treating rules and facts as data
- D.9. Miscellaneous predicates
- D.10. Definable predicates
- D.11. Debugging

Appendix E: Using this book with Micro-Prolog

Appendix F: For further reading

Appendix G: Answers to selected exercises