INDICE

Preface	VII
Parte I **Artificial intelligence: its roots and scope** Artificial intelligence and attempted definition	1
1 AI: history applications 1.1 From Eden to ANIAC: attitudes toward intelligence, knowledge., and human artifice	3
1.1.1 Historical foundations	4
1.1.2 The development off logic	7
1.1.3 The Turing test	10
1.2 Overview of AI applications areas	13
1.2.1 Game playing 1.2.2 Automated reasoning and theorem proving	14
1.2.3 Expert systems	15
1.2.4 Natural language understanding and semantic modeling	17
1.2.5 Modeling human performance	18
1.2.6 Planning and robotics	19
1.2.7 Languages and environment for AI 1.2.8 Machine learning	20
1.2.9 neural networks or parallel distributed processing (PDP)	21
1.2.10 AI and philosophy	22
1.3 Artificial intelligence A summary	23
1.4 Epilogue and references	24
1.5 Exercises	25
Part II **Artificial intelligence as representation and search**	29
Knowledge representation	30
Proving solving as search	36
2 The predicate calculus 2.0 Introduction 2.1 The propositional calculus 2.1.1 Symbols ands sentences	41
2.1.2 The semantics the propositional calculus	43
2.2 The predicate calculus 2.2.1 The syntax of predicate and sentences	46
2.2.2 A semantics for the predicate and sentences	52
2.3. Using inferences rules to produce predicate calculus expressions 2.3.1 Inferences rules	57
2.3.2 Unification	61
2.3.3 A unification example	65
2.4 Application: a logic-based financial advisor	67
2.5 Epilogue and references	72
2.6 Exercises	73
3 Structures and strategies for state space search 3.0 Introduction	75
3.1 Graph theory 3.1.1 Structures for state space search	78
3.1.1 State space representation of problems	80

3.2 Strategies for state space search 3.2.1 Data-driven and goal-driven search	86
3.2.2 IMPLEMNTING GRAPH SARCH	89
3.2.3 Depth-first and breadth-first search	92
3.2.4 Depth-first search with iterative deepening	99
3.3 Using the state space to represent reasoning with the predicate calculus 3.3.1 State space description of a logical system	100
3.3.2 And/or graphs	101
3.3. Further examples and applications	104
3.4 Epilogue and references	113
3.5 Exercises	114
4 Heuristic search 4.0 Introduction	116
4.1 An algorithm for heuristic search 4.1.1 Implementing Best-first search	120
4.1.2 Implementing heuristic evaluation functions	123
4.1.3 Heuristic search and experts systems	130
4.2 Admissibility, monotonicity, and informedness	131
4.2.1 Admissibility measures	132
4.2.2 Monotonicity	134
4.2.3 When one Heuristic is better: more informed Heuristic	135
4.3 Using Heuristic in games 4.3.1 The minimax procedure on exhaustively searchable graphs	137
4.3.2 Minimaxing to fixed ply depth	139
4.3.3 The alpha-beta procedure	142
4.4 Complexity issues	146
4.5 Epilogue and references	148
4.6 Exercises	149
5 Control and implementation of state space search 5.0 Introduction	152
5.1 Recursion-based search 5.1.1 Recursion	153
5.1.2 Recursive search	154
5.2 Pattern-directed search	156
5.3 Production systems 5.3.1 Definition and history	163
5.3.2 Examples of production systems	167
5.3.3 Control of search in production systems	172
5.3.4 Advantages of production systems for AI	177
5.4 Predicate calculus and planning	179
5.5 The blackboard architecture for problem solving	187
5.6 Epilogue and references	190
5.7 Exercises	191
Parte III **Languages for AI problem solving**	195
Languages, understanding, and levels of abstraction	196
Requirements for AI languages	198
The primary AI languages: LISP and PROLOG	205

PROLOG	206
LISP	207
Selecting and implementation language	208
6 An introduction to PROLOG 6.0 Introduction	210
6.1 Syntax for predicate calculus programming 6.1.1 Representing facts and rules	211
6.1.2 Creating, changing, and monitoring the PROLOG environment	215
6.1.3 Recursion-based search in PROLOG	216
6.1.4 Recursive search in PROLOG	219
6.1.5 The use of cut to control search in PROLOG	222
6.2 Abstract data types (ADTs) in PROLOG 6.2.1 The ADT stack	224
6.2.2 The ADT queue	225
6.2.3 The ADT priority queue	226
6.2.4 The ADT set	227
6.3 A production system example in PROLOG	228
6.4 Designing alternative search strategies 6.4.1 Depth-first search using the closed list	234
6.4.2 Breadth-first search in PROLOG	235
6.4.3 Best-first search in PROLOG	237
6.5 A PROLOG planner	239
6.6 PROLOG: toward a nonprocedural computing language	242
6.7 Epilogue and references	248
6.8 Exercises	249
7 LISP 7.0 Introduction	252
7.1 LISP: A brief overview 7.1.1 Symbolic expression, the syntactic basis of LISP	253
7.1.2 Control LSIP evaluation: quote and eval	257
7.1.3 Programming in LISP: crating new function	258
7.1.4 Program control in LISP: conditions and predicates	259
7.1.5 Functions, lists, and symbolic computing	262
7.1.6 Lists as recursive structure	264
7.1.7 Nested lists, structure and car/cdr recursion	267
7.1.8 Functional programming, side effects, set, and let	270
7.1.9 Data types in common LISP	275
7.2 Search algorithms in LSIP: a functional approach to the farmer, wolf, goat, and cabbage problem	276
7.3 Higher-order functions and pro9cedural abstraction 7.3.1 Maps and filters	281
7.3.2 Functional arguments and lambda expressions	284
7.4 Search strategies in LSIP 7.4.1 Breadth-first and depth-first search	285
7.4.2 Best-first search	288
7.5 Pattern matching in LISP	289
7.6 A recursive unification function 7.6.1 Implementing the unification algorithm	291
7.6.2 Implementing substitution sets using association lsit	293

7.7 Interpreters and embedded languages	295
7.8 Epilogue and references	298
7.9 Exercises	299
Parte IV **Representations for knowledge-based systems**	303
8 Rule-based expert systems 8.0 Introduction	308
8.1 Overview of expert systems technology 8.1.1 Design of rule-based expert systems	310
8.1.2 Selecting a problem for expert systems development	312
8.1.3 The knowledge engineering process	314
8.1.4 Conceptual models and their role in knowledge acquisition	317
8.2 A framework for organizing and applying human knowledge 8.2.1 Production systems roles, and the expert system architecture	320
8.2.2 Explanation and transparency	323
8.2.3 Heuristic and control in expert systems 8.3 Managing uncertainty in expert systems 8.3.1 Introduction	326
8.3.2 Bayesian probability theory	328
8.3.3 The Stanford certainty factor algebra	329
8.3.4 Nonmonotonic logic and reasoning with beliefs	332
8.3.5 Fuzzy logic, dempter/shafer, and other approaches to uncertainty	333
8.3 MYCIN: A case study 8.4.1 Introduction	334
8.4.2 Representation of rules and facts	335
8.4.3 MYCIN diagnosing and Illness	337
8.4.4 Evaluation of expert systems	342
8.4.5 Knowledge acquisition and the teiresias knowledge-base editor	344
8.5 Epilogue and references	349
8.6 Exercises	350
9 Knowledge representation 9.0 Knowledge representation languages	352
9.1 Issues in Knowledge representation	354
9.2 A survey of networks representations 9.2.1 Associations theories of meaning	356
9.2.2 Early work in semantic nets	360
9.2.3 Standardization of network relationships	362
9.3 Conceptual graphs a network representation language 9.3.1 Introduction to conceptual graphs	368
9.3.2 Types, individuals, and names	369
9.3.3 The type hierarchy 9.3.4 Generalization and specialization	372
9.3.5 Propositional nodes	376
9.3.6 Conceptual graphs and logic	377
9.4 Structured representation 9.4.1 Frames	378
9.4.2 Scripts	383
9.5 Type hierarchies, inheritance and exception handling	386
9.6 Further problems in knowledge representation	389

9.7 Epilogue and references	392
9.8 Exercises	393
10 Natural language 10.0 Role of in knowledge understanding	396
10.1 The Natural language problem 10.1.1 Introduction	398
10.1.2 Stages f languages analysis	399
10.2 Syntax 10.2.1 Specification and parsing sunning context-free grammars	401
10.2.2 Transition network parsers	403
10.2.3 The Chomsky hierarchy and context-sensitive grammars	408
10.3 Combining syntax and semantic in ATN parsers	410
10.3.1 Augmented transition network parsers	411
10.3.2 Combining syntax and semantic	413
10.3 Natural language applications 10.4.1 Story understanding and question answering	419
10.4.2 A data base fromt end	420
10.5 Epilogue and references	424
10.6 Exercises	425
11 Automated reasoning 11.0 Introduction to weak methods in theorem proving	427
11.1 The general problem solver and difference tables	428
11.2 Resolution theorem proving 11.2.1 Introduction	433
11.2.2 Producing the clause form for resolution refutations	436
11.2.3 The binary resolution proofs procedure	440
11.2.4 Strategies and simplification techniques for resolution	445
11.2.5 Answer extraction from resolution refutations	449
11.3 Further issues in the design of automated reasoning programs 11.3.1 Uniform representations for weak methods solutions	453
11.3.2 Alternative inference rules	456
11.3.3 Search strategies and their use	458
11.4 Epilogue and references	459
11.5 Exercises	460
12 Machine learning 12.0 Introduction	462
12.1 A framework for learning	465
12.2 Version space search 12.2.1 Generalization operators and the concept space	471
12.2.2 The candidate elimination algorithm	472
12.2.3 LEX: inducing search heuristics	479
12.2.4 Evaluating candidate elimination	482
12.3 The ID3 decision three induction algorithm	483
12.3.1 top-down decision three induction	486
12.3.2 Information theoretic test selection	487
12.3.3 Evaluating ID3 12.4 Inductive bias and learnability	490
12.4.1 Inductive bias	491
12.4.2 The theory of learnability	493

12.5 Knowledge and learning	495
12.5.1 Meta-DENDRAL	496
12.5.2 Explanation-based learning	497
12.5.3 EBL and Knowledge level learning	501
12.5.4 Analogical learning	502
12.5.5 Case based reasoning	505
12.6 Unsupervised learning	506
12.6.1 Discovery and Unsupervised learning	507
12.6.2 Conceptual clustering	508
12.6.3 COBWEB and the structures of taxonomic knowledge	510
12.7 Parallel-distributed pr9ocessing	516
12.71 Foundations of neural networks	518
12.7.2 The delta rule	522
12.7.3 Backpropagation	523
12.7.4 NET talk	526
12.8 Genetic algorithms	527
12.8.1 The Genetic algorithms	528
12.8.2 Evaluating the Genetic algorithms	529
12.9 Epilogue and references	531
12.10 Exercises	532
Part V **Advanced AI programming techniques** AI languages and Meta-interpreters	535
Object-oriented programming	536
Hybrid environment	537
A Hybrid example	538
13 Advanced representation in PROLOG 13.0 Introduction	541
13.1 PROLOG tools: meta-predicates, types, and unification 13.1.1 Meta logical predicates	542
13.1.2 Types in PROLOG	543
13.1.3 Unifications, the engine for predicate matching and evaluation	546
13.2 Meta-interpreters in PROLOG 13.2.1 An introduction to meta- interpreters in PROLOG in PROLG	550
13.2. Shell for a rule-based expert system	553
13.2.3 Semantic nets in PROLOG	562
13.2.4 Frames and schemata in PROLOG	564
13.3 Natural language understanding in PROLOG 13.3.1 Introduction	567
13.3.2 A recursive descent semantic net parser	568
13.4 Version space search in PROLOG	571
13.4.1 The feature vector representation of concepts and instances	572
13.4.2 Specific to general search	573
13.4.3 Candidate elimination	576
13.5 Explanation-based learning in PROLOG	579
13.6 PROLOG and programming with logic 13.6.1 Introduction	582
13.6.2 Logic programming and PROLOG	583
13.6.3 Parallel PROLOG and an alternatives semantics	588

13.7 Epilogue and references	591
13.8 Exercises	592
14 Advanced LISP programming techniques for artificial intelligence 14.0 Introduction abstraction and complexity	595
14.1 Logic programming in LISP	596
14.1.1 A simple Logic programming language	597
14.1.2 Streams and stream processing	599
14.1.3 A Streams-based Logic programming interpreter	601
14.2 Streams- and delayed evaluation	605
14.3 An experts system shell in LISP 14.3.1 Implementing certainty factors	610
14.3.2 Architecture of LISP-Schell	611
14.3.3 User queries and working memory	614
14.3.4 Classification using LISP-Schell	615
14.4 Network representation and inheritance	617
14.4.1 Representing semantic nets in LISP	618
14.4.2 Implementing inheritance	620
14.5 The ID3 induction algorithm	621
14.5.1 Defining structure using defstruct	622
14.5.2 Representing objects and properties	624
14.5.3 Data structures in ID3	626
14.5.4 Implementing ID3	628
14.5.5 Learning classification using build-tree	633
14.5.6 The object-oriented approach to program structure	634
14.6 Epilogue and references	635
14.7 Exercises	636
15 Objects, messages, and hybrid expert system design 15.0 Introduction	638
15.1 Object-oriented knowledge representation 15.1.1 Objects and abstraction	639
15.1.2 Benefits of object- oriented programming	646
15.1.3 Object-oriented knowledge bases	647
15.2 LSIP and object- oriented programming 15.2.1 OOPS: a simple object- oriented programming language	649
15.2.2 Implementing OOPS in LISP	652
15.2.3 An object- oriented simulation using OOPS	656
15.2.4 Evaluating OOPS 15.3 The common LISP object system	660
15.3.1 Defining classes and instances in CLOS	661
15.3.2 Defining generic functions and methods	663
15.3.3 Inheritance in CLOS	665
15.3.4 Advanced features of CLOS 15.3.5 Implementing a thermostat simulation	667
15.3 Object- oriented programming and concurrency in PROLOG 15.4.1 Introduction	671
15.4.2 Object in concurrent PROLOG	672
15.4 Hybrid expert system tools 15.5.1 Hybrid environments: integrating objects, rules, and procedures	680

15.5.2 Facets, demons, and graphics in hybrid environments	681
15.6 Epilogue and references	682
15.7 Exercises	683
Part IV **Epilogue** **16 Artificial intelligence as empirical enquiry**	687
16.0 Introduction	688
16.1 Artificial intelligence a revised definition	689
16.2 Cognitive science: and overview	694
16.2.1 The analysis of human performance	695
16.2.2 The production system and human cognition	696
16.3 Representational models for intelligence: issues and directions	699
16.4 Epilogue and references	703