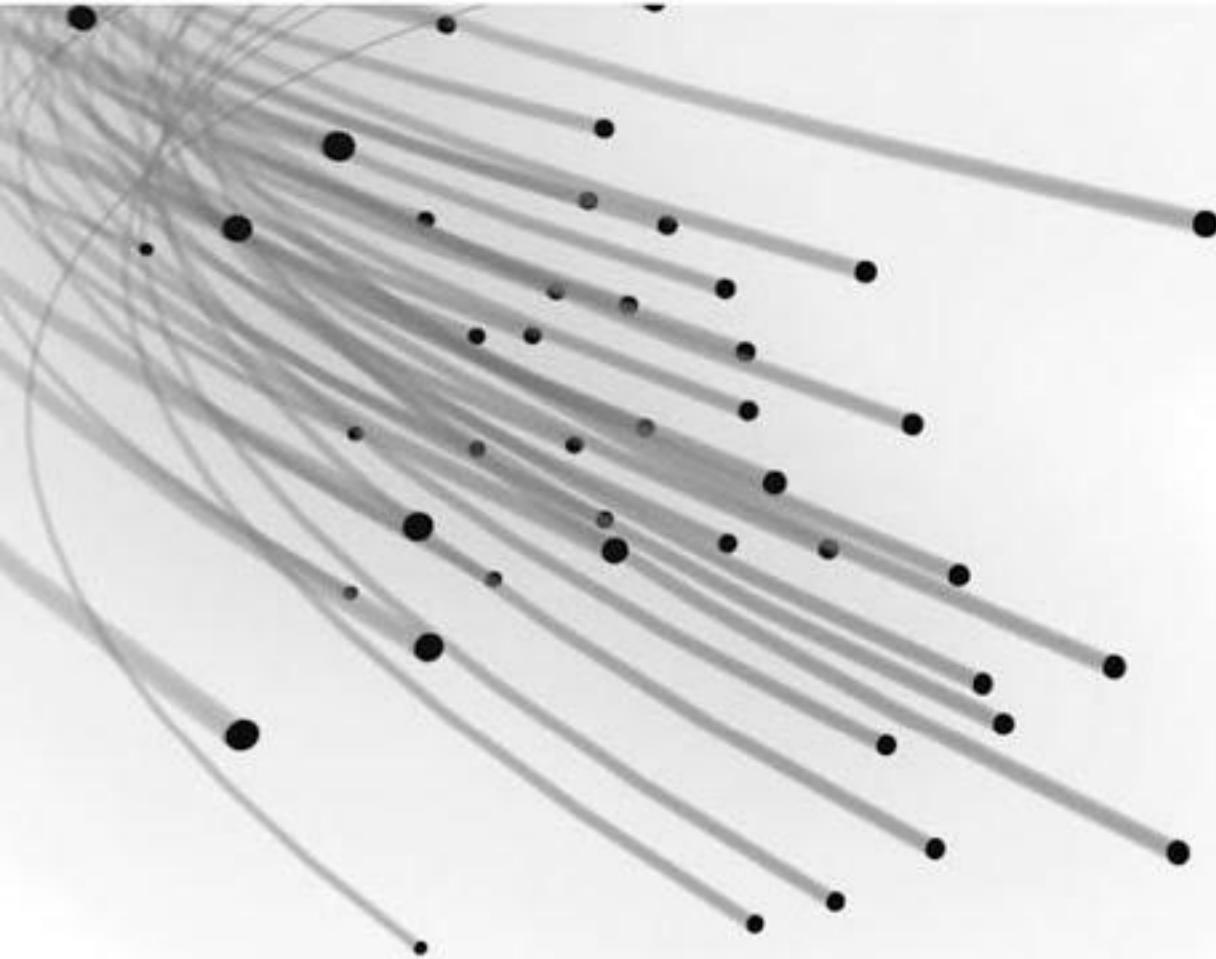


# Intelligent Systems

Principles,  
Paradigms, and  
Pragmatics



• Robert J. Schalkoff



# Intelligent Systems

Principles,  
Paradigms, and  
Pragmatics

Robert J. Schalkoff  
Clemson University



JONES AND BARTLETT PUBLISHERS

*Sudbury, Massachusetts*

BOSTON

TORONTO

LONDON

SINGAPORE

Материал, защищенный авторским правом

*World Headquarters*

Jones and Bartlett Publishers  
40 Tall Pine Drive  
Sudbury, MA 01776  
978-443-5000  
[info@jpub.com](mailto:info@jpub.com)  
[www.jpub.com](http://www.jpub.com)

Jones and Bartlett Publishers  
Canada  
6339 Ormendale Way  
Mississauga, Ontario L5V 1J2  
Canada

Jones and Bartlett Publishers  
International  
Barb House, Barb Mews  
London W6 7PA  
United Kingdom

Jones and Bartlett's books and products are available through most bookstores and online book-sellers. To contact Jones and Bartlett Publishers directly, call 800-832-0034, fax 978-443-8000, or visit our website [www.jpub.com](http://www.jpub.com).

Substantial discounts on bulk quantities of Jones and Bartlett's publications are available to corporations, professional associations, and other qualified organizations. For details and specific discount information, contact the special sales department at Jones and Bartlett via the above contact information or send an email to [specsales@jpub.com](mailto:specsales@jpub.com).

Copyright © 2011 by Jones and Bartlett Publishers, LLC

All rights reserved. No part of the material protected by this copyright may be reproduced or utilized in any form, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the copyright owner.

**Production Credits:**

Publisher: David Pallai

Acquisitions Editor: Timothy Anderson

Editorial Assistant: Melissa Potter

Production Director: Amy Rose

Senior Marketing Manager: Andrea DeFronzo

V.P., Manufacturing and Inventory Control: Therese Connell

Composition: Northeast Compositors, Inc.

Cover and Title Page Design: Kristin E. Parker

Cover and Title Page Image: © Harper/ShutterStock, Inc.

Printing and Binding: Malloy, Inc.

Cover Printing: Malloy, Inc.

**Library of Congress Cataloging-in-Publication Data**

Schalkoff, Robert J.

Intelligent systems : principles, paradigms, and pragmatics / Robert J. Schalkoff.

p. cm.

Includes index.

ISBN-13: 978-0-7637-8017-3 (hardcover)

ISBN-10: 0-7637-8017-0 (ibid.)

1. Intelligent agents (Computer software) 2. Intelligent control systems. 3. Expert systems (Computer science) 4. Artificial intelligence. I. Title.

QA76.76.I58S323 2009

006.3—dc22

2009027414

# Contents

## Preface

xxv

<b>1</b>	<b>Introduction to Intelligent Systems</b>	<b>1</b>
1.1	What Is Intelligent Systems (IS)? . . . . .	1
1.1.1	Key Topics or Themes . . . . .	1
1.1.2	IS: A Practical Definition? . . . . .	2
1.1.3	Some Significant Preliminary Questions . . . . .	2
1.1.4	IS: A Psychology-Based Viewpoint . . . . .	3
1.1.5	Two Approaches to IS Development . . . . .	4
1.2	Key Events and a History of Artificial Intelligence/Intelligent Systems (AI/IS) . . . . .	5
1.2.1	A Timetable of Key Events . . . . .	5
1.2.2	Grand Research Challenges in Computer Science and Engineering Related to IS	7
1.3	An Elusive Model Yields an Elusive Goal . . . . .	9
1.3.1	The Mind and (Versus?) the Brain . . . . .	9
1.3.2	Puzzle Solving (Riddles) and Intelligence . . . . .	10
1.3.3	Data Versus Knowledge . . . . .	11
1.3.4	Semantic Computing . . . . .	11
1.3.5	IS and Game Theory . . . . .	11
1.4	Distinguishing Characteristics of IS Problems . . . . .	12
1.5	The Turing Test . . . . .	14
1.6	Where Is the Field of IS Going? . . . . .	14
1.7	Quantitative Tools for IS Development . . . . .	15
1.8	Organization of the Text . . . . .	16
1.9	Practice or Introductory Exercises . . . . .	17
<b>2</b>	<b>First Steps in IS: Representation, Ontologies, and Obtaining Expertise</b>	<b>19</b>
2.1	Representation in IS . . . . .	19
2.1.1	Knowledge Representation and Related Definitions . . . . .	19
2.1.2	Human Reasoning: Classes, Categories, Concepts, and Chunking . . . . .	19
2.1.3	Declarative Versus Procedural Representations . . . . .	20
2.2	Tools for Representation . . . . .	20

2.2.1	Discrete Mathematics: Using Relations and Properties . . . . .	20
2.2.2	Digraphs to Semantic Nets . . . . .	21
2.2.3	Semantic Net Examples . . . . .	21
2.3	The Semantic Web: Concepts and Relationships, Not Just Data or Words . . . . .	22
2.4	Structuring the Representation: Frames and Object-Oriented Representations . . . . .	23
2.4.1	Frame Slots, “is-a,” and Inheritance . . . . .	24
2.4.2	Object-Oriented Approaches . . . . .	24
2.5	Ontologies: Concept and Tools . . . . .	26
2.5.1	Background and Motivation . . . . .	26
2.5.2	What Is an Ontology and Why Do I Care? . . . . .	26
2.5.3	Ontologies and Knowledge Bases . . . . .	27
2.5.4	Standardization of Ontology Formats . . . . .	27
2.6	The Protege Ontology Editor . . . . .	30
2.6.1	Protege Examples . . . . .	31
2.6.2	Alternate Representation Formats Using Protege . . . . .	33
2.7	Ontological Reasoning and “Common Sense” Implementations . . . . .	33
2.7.1	CYC (OpenCYC) . . . . .	33
2.7.2	Common Sense (MIT) . . . . .	35
2.8	Expert Systems . . . . .	35
2.8.1	What Is an Expert System? . . . . .	35
2.8.2	Rule-Based Implementations . . . . .	38
2.8.3	How Long Does It Take to Become an Expert? . . . . .	38
2.8.4	The Appeal of Expert Systems . . . . .	39
2.8.5	The Expert System Knowledge Engineering Process . . . . .	39
2.8.6	Production Systems for Expert System Implementation . . . . .	42
2.9	Exercises . . . . .	42
<b>3</b>	<b>Search and Computational Complexity in IS</b> . . . . .	<b>45</b>
3.1	Why Do We Care about IS Problem Computational Complexity? . . . . .	45
3.2	The Concept of a System State Space . . . . .	45
3.2.1	System State Representation . . . . .	45
3.2.2	The System State Space . . . . .	46
3.2.3	Satisfying Goals via State Manipulation with Operators . . . . .	47
3.2.4	Nondeterminism: Unreachable States . . . . .	47
3.3	Computational Complexity . . . . .	48
3.3.1	Complexity Functions . . . . .	48
3.3.2	An Example of the State Space and IS Problem Complexity . . . . .	49
3.4	Search . . . . .	51
3.4.1	To Search or Not to Search . . . . .	51
3.4.2	Problem Representation . . . . .	51
3.4.3	Definition of a Search Problem ( $P$ ) . . . . .	52
3.4.4	State-Space Graphs . . . . .	52
3.4.5	Search Costs and Heuristics . . . . .	53

3.4.6	Graphical Representations for Search . . . . .	53
3.4.7	Example: A Graphical Representation of State Propagation . . . . .	53
3.4.8	From Graph to Tree Representations . . . . .	55
3.4.9	Example: Another Graphical Representation of Search . . . . .	55
3.5	Searching the Problem State Space . . . . .	57
3.5.1	Example of Search and State-Space Paths . . . . .	57
3.5.2	The Explosion of Operator Choices . . . . .	57
3.6	Heuristics . . . . .	59
3.6.1	What Is a Heuristic? . . . . .	59
3.6.2	Definitions . . . . .	59
3.6.3	Applications . . . . .	59
3.6.4	Sample IS Heuristics . . . . .	60
3.6.5	Why Use Heuristics? . . . . .	60
3.6.6	The Power, Application, and Properties of Heuristics . . . . .	61
3.6.7	Machine Learning of Heuristics . . . . .	61
3.7	Informed Versus Uninformed (“Blind”) Search . . . . .	62
3.8	The Computational Cost of Search . . . . .	62
3.8.1	The Computational Complexity of Forward Propagation of States . . . . .	63
3.8.2	Backward State Propagation . . . . .	63
3.9	Common Search Algorithms . . . . .	65
3.9.1	Brute Force (“Uninformed”) Algorithms . . . . .	65
3.9.2	Systematic Search Strategies . . . . .	65
3.10	Other Search Approaches (Informed Search) . . . . .	66
3.11	Exercises . . . . .	68
<b>4</b>	<b>Constraint Satisfaction Problems, Part 1</b>	<b>71</b>
4.1	Nonstructural Constraint Satisfaction Problems (CSPs) . . . . .	71
4.1.1	A Simple Introductory Example: Map Coloring . . . . .	71
4.1.2	Another CSP Example: Numerical Constraint Satisfaction . . . . .	72
4.1.3	Formal Definition . . . . .	73
4.2	CSPs as Propositional Satisfiability (SAT) Problems . . . . .	73
4.3	SAT: Definition, Properties, and Solutions . . . . .	74
4.3.1	$k$ -SAT . . . . .	74
4.3.2	SAT Solutions . . . . .	74
4.3.3	SAT Properties . . . . .	74
4.4	A (Somewhat) Obvious Solution to CSPs: Generate-and-Test (GAT) . . . . .	75
4.5	Constraint Programming and a Role for Prolog in CSPs . . . . .	75
4.6	The Labeling (Assignment) Problem . . . . .	76
4.6.1	Labeling Complexity . . . . .	76
4.6.2	Image Labeling Example . . . . .	76
4.6.3	CSP Solutions in Prolog . . . . .	77
4.6.4	Adding Unary Constraints . . . . .	78
4.6.5	Adding Global Constraints . . . . .	79

4.6.6	Adding Optimization to the Formulation . . . . .	82
4.7	The Map Coloring Problem . . . . .	85
4.7.1	Background and History . . . . .	85
4.7.2	Prolog-Based Solution Approaches . . . . .	86
4.8	Another Framework for CSP Assignment Problems and Alternative Solution Approaches . . . . .	88
4.8.1	A General Representational Framework . . . . .	88
4.8.2	Enforcing $k = 2$ Object Label Compatibility . . . . .	89
4.8.3	Iterative CSP Solutions via Constraint Propagation: Introducing Discrete Relaxation . . . . .	90
4.9	Investigating Solution Efficiencies . . . . .	98
4.9.1	Considering Search . . . . .	98
4.9.2	GAT . . . . .	99
4.9.3	GAT with Heuristic Guidance . . . . .	99
4.9.4	Backtracking . . . . .	99
4.9.5	Constraint Propagation . . . . .	100
4.10	Extension to Continuous/Probabilistic Formulations . . . . .	100
4.10.1	“Relaxing” and Extending Several Aspects of Discrete Relaxation . . . . .	100
4.10.2	An Algorithm for Propagating $P_i(\lambda)$ . . . . .	102
4.10.3	Examples and Extensions . . . . .	103
4.11	Summary . . . . .	108
4.12	A More Comprehensive CSP: Electric Power System Protection Analysis . . . . .	108
4.12.1	Background and Motivation . . . . .	108
4.12.2	Objectives . . . . .	108
4.12.3	Simple Power Protection Background . . . . .	109
4.12.4	Problem Formulation . . . . .	109
4.12.5	Prolog Description of System Topology . . . . .	110
4.12.6	Breaker Backup . . . . .	111
4.12.7	Detailed Prolog Description Derivation . . . . .	111
4.12.8	Listing of Overall Prolog Program . . . . .	117
4.12.9	Sample Results . . . . .	120
4.12.10	Summary: Expert System for Electric Power System Protection Interpretation . . . . .	121
4.13	Extensions . . . . .	121
4.14	Exercises . . . . .	121
<b>5</b>	<b>CSPs, Part 2: Structural Approaches Leading to Natural Language Understanding and Related Topics</b>	<b>129</b>
5.1	Introduction . . . . .	129
5.2	What Types of Problems Have Structural Constraints? . . . . .	129
5.3	Natural Language and Structure . . . . .	132
5.4	Early NL Understanding Efforts . . . . .	132
5.4.1	McCarthy’s Proposal . . . . .	132
5.4.2	ELIZA . . . . .	133

5.5	Natural Language Understanding: Grammars . . . . .	136
5.5.1	Example: Sentence Formation As Productions (Rewrite Rules) . . . . .	136
5.5.2	The Phase Structure (String) Grammar . . . . .	137
5.5.3	$L(G)$ . . . . .	138
5.5.4	Grammar Modes . . . . .	138
5.5.5	Grammar Types and Productions . . . . .	138
5.5.6	Syntax, Semantics, and Context . . . . .	139
5.5.7	Graphical Aids . . . . .	139
5.6	Sentence Generation and Recognition Using Prolog . . . . .	140
5.6.1	The Initial Grammar Model . . . . .	140
5.6.2	Grammar G1 . . . . .	140
5.6.3	Sample Results for a Simple Grammar . . . . .	141
5.6.4	Showing Structure and Extending the Language . . . . .	142
5.6.5	Enforcing Meaningful Semantics . . . . .	144
5.7	From a Simple Parse to Structure to Meaning . . . . .	148
5.8	Feature Structure-Based Representation and Manipulation . . . . .	149
5.8.1	Motivations . . . . .	149
5.8.2	Feature Structures (FS) and Basic Properties . . . . .	149
5.8.3	Feature Structure-Based Manipulation . . . . .	153
5.8.4	Feature Structure Representation Enhancements . . . . .	157
5.8.5	Rules as Feature Structures . . . . .	158
5.8.6	Lattices of Feature Structures . . . . .	158
5.8.7	Lattice Processing Directions . . . . .	159
5.9	Exercises . . . . .	161
<b>6</b>	<b>From Logic-Based Chaining to Production Systems</b>	<b>167</b>
6.1	Conceptual Background and Motivation . . . . .	167
6.2	The Role of Logic in IS Representation and Manipulation . . . . .	167
6.2.1	Inference and the Vocabulary of Logic . . . . .	168
6.2.2	Other Logic Families . . . . .	169
6.3	Logic-Based Underpinnings of Rule-Based Chaining Systems . . . . .	169
6.3.1	The Implication Connective . . . . .	169
6.3.2	An Implication-Based IS Rule Example . . . . .	170
6.3.3	Rule-Based Representation Examples . . . . .	170
6.3.4	Business Rules and Engines . . . . .	171
6.3.5	Implication and Inference: Simple Modus Ponens (MP) . . . . .	172
6.3.6	Modus Ponens with Variables . . . . .	172
6.3.7	The Concept and Process of Unification . . . . .	172
6.3.8	Resolution (Proof by Refutation): An Alternative to MP . . . . .	176
6.3.9	Mechanical Implementations of Logic Can Lead to Illogical Results . . . . .	178
6.4	Chaining, Inference Directions, and Potential Complexities in Chaining . . . . .	178
6.5	Implementing Rule-Based Chaining . . . . .	180
6.5.1	The Role and Design of the System Inference Engine . . . . .	180
6.5.2	Computational Elements of Rule-Based Chaining and Examples . . . . .	181

6.6	Extensions to the Simple Chaining Paradigm: Rule Selection, Conflict Resolution Measures, and the Conflict Set . . . . .	181
6.6.1	The Conflict Set . . . . .	181
6.6.2	Possible Conflict Resolution Strategies . . . . .	182
6.6.3	Conflict Resolution Examples . . . . .	183
6.6.4	Conflict Resolution Computational Aspects and Tradeoffs . . . . .	185
6.7	The Production System Paradigm . . . . .	185
6.7.1	Programming Paradigms and Production Systems . . . . .	185
6.7.2	A More General Viewpoint: <i>Rules Trigger Actions</i> . . . . .	185
6.7.3	Production Systems Architecture . . . . .	186
6.7.4	Features of Rule-Based Production Systems . . . . .	187
6.7.5	Production System Properties . . . . .	188
6.7.6	Decomposable Production Systems . . . . .	189
6.8	Production Systems: The Next Step . . . . .	190
6.9	Exercises . . . . .	190
<b>7</b>	<b>The C Language Integrated Production System (CLIPS)</b> . . . . .	<b>195</b>
7.1	CLIPS: Conceptual Background and Motivation . . . . .	195
7.1.1	CLIPS History . . . . .	195
7.1.2	CLIPS Structure . . . . .	195
7.1.3	CLIPS Nomenclature . . . . .	196
7.1.4	Simplified CLIPS Execution Cycle . . . . .	196
7.1.5	CLIPS Documentation . . . . .	196
7.1.6	Introductory CLIPS Examples . . . . .	197
7.1.7	Logging a CLIPS Session . . . . .	201
7.1.8	CLIPS Help . . . . .	202
7.2	Selected Syntax, Constructs, and Examples . . . . .	202
7.2.1	Predominant CLIPS Constructs . . . . .	202
7.2.2	A Third Example—Prelude to an Expert System . . . . .	210
7.3	CLIPS Conflict Resolution and the <i>agenda</i> . . . . .	210
7.3.1	Controlling Production System Computation . . . . .	210
7.3.2	The <i>agenda</i> . . . . .	212
7.3.3	User-Defined Rule Salience . . . . .	213
7.3.4	Forming the Conflict Set ( <i>agenda</i> ) . . . . .	215
7.3.5	Available CLIPS Conflict Resolution Strategies . . . . .	215
7.3.6	Conflict Resolution Examples . . . . .	216
7.4	Another Expert System Application Example: “Tenure” . . . . .	216
7.4.1	An Application Domain . . . . .	216
7.4.2	The Development Process . . . . .	219
7.4.3	Initial CLIPS Expert System Development Example . . . . .	219
7.4.4	Enhanced CLIPS Development of the Previous Example . . . . .	221
7.4.5	A CLIPS Planning Example: The Monkey and Bananas (MAB) Problem . . . . .	225
7.5	CLIPS Efficiency and the Rete Algorithm . . . . .	226
7.5.1	Motivation . . . . .	226

7.5.2	Computational Complexity . . . . .	227
7.5.3	The Computational Model . . . . .	227
7.5.4	The “Traditional” Viewpoint of Production System Operation . . . . .	227
7.5.5	Rete (I) Background . . . . .	229
7.5.6	Productions and <code>WM</code> Data: Who’s In Charge? . . . . .	229
7.5.7	How Is Rete Implemented? . . . . .	231
7.5.8	Exploring the Rete Algorithm with CLIPS . . . . .	234
7.5.9	Writing Efficient CLIPS Programs . . . . .	236
7.5.10	Beyond Rete . . . . .	238
7.6	Embedded CLIPS . . . . .	238
7.6.1	Embedding a Production System . . . . .	238
7.6.2	Sample CLIPS Embedding (Using C) . . . . .	238
7.7	Descendants of CLIPS . . . . .	240
7.7.1	JESS . . . . .	240
7.7.2	Clips in Java . . . . .	242
7.7.3	Soar . . . . .	242
7.8	CLIPS Web Resources and Downloads . . . . .	242
7.9	Exercises . . . . .	242

## 8 Extended Production System Representation and Manipulation Approaches, Including Agents 253

8.1	Modular Production System Development Approaches . . . . .	253
8.2	Implementing Modular Representations in CLIPS . . . . .	254
8.2.1	Introduction . . . . .	254
8.2.2	Module Definition Syntax . . . . .	254
8.2.3	Specifying the Module for a Construct . . . . .	254
8.2.4	Examples of Module Specifications: Different Knowledge Domains . . . . .	254
8.2.5	Module Visibility . . . . .	256
8.2.6	Example: Changing Focus . . . . .	257
8.3	Object-Oriented (OO) Representation, Production Systems, and CLIPS COOL . . . . .	258
8.3.1	COOL Class Syntax . . . . .	258
8.3.2	Sample Class Hierarchy in COOL . . . . .	258
8.3.3	Examining the COOL Classes . . . . .	259
8.3.4	Creating Instances . . . . .	262
8.3.5	Integrating Classes, Instances, and Productions in COOL . . . . .	262
8.4	Protégé, OO-Based Ontologies, CLIPS, and COOL . . . . .	266
8.4.1	Protégé Representation Files . . . . .	266
8.4.2	Graphical Representation of the Sample Ontology . . . . .	266
8.4.3	Importing <code>*.pont</code> Files into CLIPS . . . . .	266
8.4.4	Importing <code>*.pins</code> Files into CLIPS . . . . .	266
8.4.5	Loading the Protégé Ontology into CLIPS . . . . .	270
8.4.6	Combining CLIPS Productions with the Protégé Sample Ontology . . . . .	270
8.4.7	Instance-Set Queries in COOL . . . . .	271
8.4.8	Protégé, fuzzyCLIPS, and COOL . . . . .	272

8.5	IS Representation/Knowledge Base Consistency . . . . .	272
8.5.1	Redundancy . . . . .	272
8.5.2	Conflicts . . . . .	272
8.5.3	Subsumed or Subordinate Rules . . . . .	272
8.5.4	Unnecessary Conditions . . . . .	273
8.5.5	Circular Rules . . . . .	273
8.6	Nonmonotonic Logic . . . . .	273
8.6.1	Definition . . . . .	273
8.6.2	Examples of Situations Involving Nonmonotonic Reasoning . . . . .	274
8.6.3	Soar and Nonmonotonic Reasoning . . . . .	274
8.6.4	Nonmonotonic Reasoning Formalisms . . . . .	274
8.7	Reasoning with Time (Temporal Logics and Operators) . . . . .	275
8.7.1	Characteristics of Temporal Reasoning . . . . .	275
8.7.2	Subdivisions of Time . . . . .	275
8.7.3	Time Intervals and Points . . . . .	276
8.7.4	Possible Implementations of RWT . . . . .	276
8.8	IS Collaboration: Blackboards . . . . .	279
8.8.1	Blackboard Control . . . . .	280
8.8.2	Typical Sequence of Blackboard Operations . . . . .	280
8.9	From Production Systems to Distributed, Autonomous Agents . . . . .	281
8.9.1	What Is an Agent? . . . . .	281
8.9.2	Definitions . . . . .	281
8.9.3	An Agent Example: NL Understanding . . . . .	282
8.9.4	An Agent Example: Network Monitoring . . . . .	282
8.9.5	Agent Characteristics . . . . .	283
8.10	Basic Agent Structures and Types of Agents . . . . .	283
8.10.1	Generic Structure . . . . .	283
8.10.2	Agent Types . . . . .	284
8.10.3	BDI Agents . . . . .	284
8.10.4	Theoretical Components of an Agent . . . . .	285
8.10.5	Agents and Rule-Based Systems . . . . .	285
8.10.6	Living Organisms As Agents . . . . .	285
8.10.7	Potential Applications for Agents . . . . .	285
8.10.8	Multiple, Mobile Agents and Distributed Computing . . . . .	286
8.10.9	Software Agents As a Design Paradigm . . . . .	287
8.10.10	Agent Software Standards . . . . .	287
8.10.11	Tools for Agent-Based Development . . . . .	287
8.10.12	Merging Multiple Agents and Blackboards . . . . .	288
8.11	References . . . . .	288
8.12	Exercises . . . . .	288
<b>9</b>	<b>Soar</b>	<b>293</b>
9.1	Soar Background . . . . .	293
9.1.1	Soar History . . . . .	293

9.1.2	IS Design and Implementation Using Soar . . . . .	294
9.1.3	Significant Elements of Soar Philosophy and Pragmatics . . . . .	295
9.2	Soar Syntax, Computational Architecture, and Examples . . . . .	296
9.2.1	Soar Memory Architecture . . . . .	296
9.2.2	States and Working Memory ( <i>wm</i> ) in Soar . . . . .	297
9.2.3	Variables in Soar . . . . .	297
9.2.4	Comments and Constants . . . . .	298
9.2.5	Hierarchical Representations and Objects . . . . .	298
9.2.6	Production Memory ( <i>pm</i> ) and Production Syntax . . . . .	298
9.2.7	Getting Started with a (Very) Simple Example . . . . .	299
9.2.8	Arithmetic in Soar . . . . .	303
9.3	Digging Deeper into Soar . . . . .	303
9.3.1	Long-Term Knowledge in Soar . . . . .	303
9.3.2	Simplified Soar Execution Cycle . . . . .	304
9.3.3	Preference Memory . . . . .	305
9.3.4	Operator Preference . . . . .	305
9.3.5	SOAR Operator Preference Syntax and Semantics . . . . .	305
9.3.6	Steps in Soar Operator Preference Resolution . . . . .	306
9.3.7	States and State Elaboration . . . . .	307
9.3.8	General Operator Definition . . . . .	307
9.3.9	I-Support Versus O-Support . . . . .	307
9.3.10	Example #2: Operator-Based State Initialization . . . . .	308
9.3.11	Soar Operator Life Cycle . . . . .	308
9.3.12	Example #3: Operator Preference Use . . . . .	309
9.3.13	Example #4: Goal State Checking . . . . .	309
9.4	Expanded View of the Soar Execution Cycle . . . . .	313
9.5	Another Soar Example: Revisiting the Image Labeling Problem . . . . .	314
9.6	An Example of Using Soar for Diagnosis . . . . .	316
9.6.1	“Lightbulb” Diagnosis Formulation . . . . .	317
9.6.2	Operator-Based Soar Formulation and Results . . . . .	317
9.7	Soar Input/Output . . . . .	321
9.7.1	Simple Text Output . . . . .	321
9.7.2	Communication Between Soar Agents and the Outside World . . . . .	321
9.8	Impasses, Subgoaling, and Learning (Chunking) . . . . .	322
9.8.1	Why Soar Is Different . . . . .	322
9.8.2	An Impasse . . . . .	324
9.8.3	Soar’s Impasse Resolution Activity . . . . .	325
9.8.4	Impasse (Tie) Example . . . . .	325
9.8.5	Handling Impasses: Available Impasse Resolution Productions in the Default Library . . . . .	327
9.8.6	Chunks and Learning . . . . .	327
9.9	Debugging Soar Representations . . . . .	327
9.9.1	The Soar Visual Debugger . . . . .	327

9.9.2	Additional Debugging Techniques . . . . .	328
9.10	Soar, Ontologies, and Other Development Aids . . . . .	329
9.10.1	Background . . . . .	329
9.10.2	Herbal . . . . .	329
9.11	Soar “Under the Hood”: So You Really Want to Develop Embedded, Autonomous Agents? . . . . .	329
9.11.1	A Note About Different Soar Platforms/Implementations . . . . .	329
9.11.2	The Soar Computational Architecture . . . . .	330
9.11.3	The Soar Kernel . . . . .	330
9.11.4	Agents and the Soar Kernel . . . . .	330
9.11.5	C++ Interfaces to Soar Using SML . . . . .	334
9.12	Soar Resources and References . . . . .	344
9.13	Exercises . . . . .	347

## **10 Representing and Manipulating Uncertainty in IS, Part1: Confidence**

	<b>Factors, Probability, Belief Networks, and Multivalued Logic</b>	<b>353</b>
10.1	Introduction . . . . .	353
10.1.1	Representing and Manipulating Uncertainty . . . . .	353
10.1.2	Expressing Uncertainty in Statements . . . . .	353
10.1.3	Approaches to Uncertainty . . . . .	354
10.2	Use and Limitations of Probability in IS . . . . .	354
10.2.1	Conditional Probability . . . . .	354
10.2.2	Bayes Rule . . . . .	355
10.2.3	Independence . . . . .	355
10.2.4	Limitations of the Probabilistic Approach . . . . .	356
10.2.5	A Simple Example of Probability Limitation Leading Up to Confidence Factors . . . . .	356
10.2.6	Joint Probability (Conjunction) Propagation . . . . .	357
10.2.7	Bayes Rule for Information Fusion . . . . .	358
10.3	From Probability to Confidence Factors . . . . .	359
10.3.1	Introduction . . . . .	359
10.3.2	Confidence Factors . . . . .	360
10.3.3	A Heuristic Approach . . . . .	360
10.3.4	Comparison of the Heuristic Approach with Probability . . . . .	360
10.3.5	Positive and Negative Reinforcement in the Heuristic Approach . . . . .	361
10.3.6	Extension of the Heuristic Approach to OR Formulations . . . . .	362
10.3.7	Confidence Factors Are Implemented in CLIPS . . . . .	362
10.4	Belief Networks . . . . .	362
10.4.1	Definitions . . . . .	362
10.4.2	BN Example and Use . . . . .	363
10.5	Multivalued Logic . . . . .	364
10.5.1	Review of Two-Valued Logic . . . . .	364
10.5.2	Extension to Multivalued Logics . . . . .	364
10.5.3	A Simple Heuristic Three-Valued Logic System Example . . . . .	364
10.5.4	Formalizing MVL: Truth Sets and Truth Values . . . . .	365

10.5.5 "Weak" Connectives in Multivalued Logic . . . . .	366
10.5.6 Using Multivalued Logic for Inference . . . . .	369
10.6 Exercises . . . . .	370
<b>11 Representing and Manipulating Uncertainty in IS, Part 2: Fuzzy Systems and FuzzyCLIPS</b>	<b>373</b>
11.1 Fuzzy Sets and Fuzzy Logic . . . . .	373
11.1.1 Examples of Fuzzy Concepts, Connotations, and Use . . . . .	373
11.1.2 Sample Fuzzy Applications . . . . .	374
11.2 Fundamental Fuzzy System Concepts . . . . .	375
11.2.1 Representing Sets with Membership Functions . . . . .	375
11.2.2 Membership Functions Resulting from Set-Theoretic Operations . . . . .	379
11.2.3 The Extension Principle . . . . .	379
11.3 General Fuzzy System Structure . . . . .	380
11.3.1 Major Components . . . . .	380
11.3.2 Linguistic Variables . . . . .	380
11.3.3 Fuzzy Antecedents and Rules . . . . .	381
11.4 Fuzzy System Design Procedures . . . . .	381
11.4.1 Principal Tasks . . . . .	381
11.4.2 Fuzzy Computational Mechanisms . . . . .	382
11.4.3 The Plethora of Fuzzy CRI and Defuzzification Strategies . . . . .	383
11.4.4 Applying Compositional Rules of Inference (CRI) . . . . .	383
11.5 A Fuzzy System Application (Control) Example . . . . .	384
11.5.1 The Plethora of Defuzzification Techniques . . . . .	385
11.5.2 Fuzzy Approach Shortcomings . . . . .	387
11.6 Exploring Uncertainty and Fuzzy Concepts with CLIPS . . . . .	387
11.6.1 Introduction to FuzzyCLIPS . . . . .	387
11.6.2 Implementing Confidence Factors in FuzzyCLIPS . . . . .	388
11.6.3 Implementing Fuzzy Sets and Reasoning in CLIPS . . . . .	391
11.7 Exercises . . . . .	413
<b>12 Planning in IS</b>	<b>417</b>
12.1 Introduction . . . . .	417
12.1.1 Path Planning . . . . .	418
12.1.2 Blocks World . . . . .	418
12.1.3 Planning Related to a Production System . . . . .	419
12.1.4 Planning Representations . . . . .	419
12.1.5 Representation Choice: Explicit Representation of Everything or Allowing Derivable Information . . . . .	419
12.1.6 The Frame Problem in Planning . . . . .	420
12.1.7 Developing a Representation . . . . .	421
12.1.8 Plan Generation As a Control Problem . . . . .	421
12.1.9 Planning Difficulties: Local Actions and Global Objectives . . . . .	422
12.1.10 Cycles . . . . .	422
12.1.11 Unreachable States . . . . .	423

12.1.12 Trade-offs . . . . .	424
<b>12.2 Representation and STRIPS . . . . .</b>	<b>424</b>
12.2.1 Operator Characterization in STRIPS . . . . .	424
12.2.2 STRIPS Blocks-World Example . . . . .	425
12.2.3 Another Simple STRIPS (Robot) Example . . . . .	426
<b>12.3 Plan Generation Algorithms . . . . .</b>	<b>426</b>
12.3.1 Generate-and-Test . . . . .	427
12.3.2 An Algorithm for FSP . . . . .	427
<b>12.4 Cataloging (Remembering) and Re-Using a Plan . . . . .</b>	<b>428</b>
12.4.1 The Triangle Table . . . . .	428
12.4.2 Example: A Triangle Table for a Blocks-Moving Problem . . . . .	430
12.4.3 Properties and Use of the Triangle Table . . . . .	430
<b>12.5 Planning, Abstraction, and Subgoals . . . . .</b>	<b>432</b>
12.5.1 Details, Details, and Defeating Search Complexity . . . . .	432
12.5.2 Planning in Abstraction Space . . . . .	433
12.5.3 Planning Using Subgoals Generated from Abstract Space . . . . .	435
<b>12.6 Parallel Actions in Planning . . . . .</b>	<b>436</b>
12.6.1 More Is Better? . . . . .	436
12.6.2 Extending Operator Representations to Multiagents or Parallel Actions . . . . .	436
<b>12.7 Planning Implementations in CLIPS . . . . .</b>	<b>438</b>
12.7.1 The Sample Problem . . . . .	438
12.7.2 The Desired Solution . . . . .	440
12.7.3 Sample CLIPS Solutions . . . . .	440
<b>12.8 Planning Implementations Using Soar . . . . .</b>	<b>449</b>
12.8.1 A Simple “Warm-Up” Problem: Using a Single Block and a $3 \times 3$ Checkerboard . . . . .	450
12.8.2 More Elaborate Planning (and Soar) Using a $4 \times 4$ Checkerboard and Two Blocks . . . . .	457
<b>12.9 Exercises . . . . .</b>	<b>488</b>
<b>13 Biologically-Inspired Computing and IS: Neural Networks (Part 1) . . . . .</b>	<b>493</b>
13.1 Conceptual Background and Motivation . . . . .	493
13.2 Relationship of IS to ANNs . . . . .	493
13.3 Biology and ANN Building Blocks . . . . .	494
13.3.1 Physical (Biological) Neurons . . . . .	494
13.3.2 Abstracting Biological Mechanisms into Artificial Unit Characteristics . . . . .	495
13.3.3 Two-Part Unit Models: Activation and Squashing . . . . .	495
13.3.4 The Sigmoid (Logistic) Squashing Function . . . . .	496
13.3.5 Sigmoid Derivative . . . . .	497
13.4 The Multilayer Feedforward Structure, Architectures, and Notation . . . . .	497
13.5 Training the MLFF Network: The Generalized Delta Rule . . . . .	501
13.5.1 Training Considerations . . . . .	501
13.5.2 Overview of the Generalized Delta Rule (GDR) . . . . .	501

13.5.3 A Formulation Introducing Delta . . . . .	504
13.5.4 Output Unit Weight Corrections . . . . .	506
13.5.5 A Modified Procedure for Hidden Units: The GDR . . . . .	507
13.6 MLFF ANN: Internal/Hidden Layers and Network Mapping Ability . . . . .	513
13.6.1 How Many Hidden Layers Are Needed? . . . . .	513
13.6.2 A Theoretical Result for Continuous Activation Functions . . . . .	514
13.7 Comprehensive MLFF ANN Training and Application Examples . . . . .	514
13.7.1 Fruit Classification . . . . .	514
13.7.2 Blocks Movement . . . . .	516
13.7.3 Learning an A/D Converter Function . . . . .	518
13.8 Resources . . . . .	519
13.9 References . . . . .	520
13.10 Exercises . . . . .	520
<b>14 Neural Networks (Part 2): Recurrent Networks and IS Applications</b>	<b>527</b>
14.1 Introduction . . . . .	527
14.2 Basic Parameters and Recurrent Network Design . . . . .	527
14.2.1 Network Parameters . . . . .	528
14.2.2 Network Dynamics . . . . .	529
14.2.3 Determining and Quantifying Network States and Behavior . . . . .	530
14.3 Weight Storage Prescription and Network Capacity . . . . .	531
14.3.1 Weight Prescriptions . . . . .	531
14.3.2 Additional Characterizations of the Storage Prescription . . . . .	532
14.3.3 Network Capacity Estimation . . . . .	533
14.4 Recurrent Network Design Procedures and Example CAM Applications . . . . .	533
14.4.1 General Design Procedure . . . . .	533
14.4.2 Another CAM Example: Design of a Simple Hopfield Network— Storing and Accessing Stable States . . . . .	534
14.4.3 CAM Example: Association of Simple Two-Dimensional Patterns . . . . .	536
14.5 Recurrent Networks: Energy Function Characterization . . . . .	536
14.5.1 Energy Analysis or “Why Hopfield Networks Work” . . . . .	538
14.5.2 Example: $d = 4$ Network Energy Function . . . . .	539
14.5.3 Summary of the Generalized Hopfield Network Equations . . . . .	540
14.6 Recurrent ANN Constraint Satisfaction and Optimization Applications . . . . .	540
14.6.1 Optimization and CSP Problems . . . . .	540
14.6.2 Mapping Constraints and Objectives into Recurrent Networks . . . . .	541
14.6.3 Other Interpretations of $E$ . . . . .	541
14.6.4 The Overall Design Process . . . . .	542
14.6.5 Unit Characteristics and Weights . . . . .	542
14.7 Example: Partitioning of Sets . . . . .	543
14.7.1 Problem Statement, Case #1 . . . . .	543
14.7.2 Problem Representation . . . . .	543
14.7.3 Constraints . . . . .	544
14.7.4 Network Energy Functions . . . . .	545

14.7.5	Solving for Weights and Biases . . . . .	546
14.7.6	Looking at $W$ . . . . .	547
14.7.7	Case #2: Optimization Measure and Associated $E$ . . . . .	547
14.8	Example: State Representation and the Traveling Salesman Problem . . . . .	548
14.8.1	Embedding Constraints in $E$ . . . . .	549
14.8.2	Mapping Constraints and Objectives into $E$ . . . . .	549
14.8.3	Unit Characteristics and Weights . . . . .	550
14.8.4	The Role of the Hopfield Unit Bias . . . . .	550
14.8.5	Discrete or Continuous Unit Models . . . . .	551
14.8.6	Sample Results . . . . .	551
14.8.7	Other Sample CSP Applications . . . . .	551
14.9	The Psychology of “Recall” and ANN Bidirectional Associative Memory (BAM) Structures . . . . .	551
14.9.1	The Order of Thought . . . . .	551
14.9.2	Discrimination, Association, and Principles of Connection (James) . . . . .	552
14.9.3	Relevant Principles of Association . . . . .	552
14.9.4	BAM Evolution from Hopfield (Totally Recurrent) Structure . . . . .	554
14.9.5	Architecture . . . . .	556
14.9.6	Training and Connections . . . . .	556
14.9.7	BAM Examples . . . . .	558
14.9.8	BAM as Partitioned Bipolar Hopfield and Other Connection Matrices . . . . .	562
14.10	References . . . . .	563
14.11	Exercises . . . . .	564
<b>15</b>	<b>Neural Networks (Part 3): Self-Organizing Systems</b>	<b>569</b>
15.1	Biological Justification . . . . .	569
15.2	Self-Organization via Clustering . . . . .	569
15.2.1	Determining “Natural Clusters” . . . . .	569
15.2.2	Clustering Similarity Measures . . . . .	570
15.2.3	Clustering Complexity . . . . .	570
15.2.4	Clustering Algorithm Parameters . . . . .	571
15.3	The $c$ -Means Algorithm . . . . .	571
15.3.1	Algorithm Description . . . . .	571
15.3.2	A Simple 1D $c$ -Means Example and Visual Interpretation . . . . .	572
15.3.3	Example: Clustering Digits (0, 1, 2, ..., 9) . . . . .	574
15.4	Self-Organizing Feature Maps (SOFM) . . . . .	574
15.4.1	Introduction . . . . .	574
15.4.2	Unit Topologies . . . . .	574
15.4.3	Defining Topological Neighborhoods (“Bubbles”) . . . . .	575
15.4.4	Network Learning Algorithm . . . . .	576
15.4.5	Network Coordinate Systems—Topological and Weight Spaces . . . . .	577
15.4.6	Algorithm Properties and Discussion . . . . .	577
15.4.7	A Simple 1-D SOFM Example . . . . .	578
15.4.8	Validation and Interpretation of the Resulting SOFM . . . . .	580

15.4.9 Another Example: 2-D SOFM Application to $11 \times 8$ Digits . . . . .	581
15.5 The Neural Gas (NG) Self-Organizing Network . . . . .	583
15.5.1 NG Algorithm . . . . .	583
15.5.2 How Is the NG Sorted Array Used? . . . . .	584
15.5.3 Engineering and Pragmatic Concerns . . . . .	584
15.5.4 NG Example . . . . .	585
15.5.5 Validation of NG Example . . . . .	585
15.5.6 Neural Gas Statistical Characterization . . . . .	587
15.6 Growing Neural Gas (GNG) . . . . .	588
15.6.1 Algorithm Description . . . . .	588
15.6.2 GNG Example . . . . .	589
15.7 "Batch" SOFM and NG Approaches . . . . .	589
15.7.1 Batch SOFM . . . . .	590
15.7.2 Batch NG . . . . .	590
15.8 References . . . . .	591
15.9 Exercises . . . . .	591
<b>16 Learning in IS</b>	<b>595</b>
16.1 Introduction to Machine Learning (ML) . . . . .	595
16.1.1 Preface . . . . .	595
16.1.2 Where Do We Integrate Learning in an IS? . . . . .	595
16.1.3 Learning about Learning . . . . .	596
16.1.4 Learning History and Psychology . . . . .	597
16.1.5 Learning Definition(s) and Examples of Automated Learning Paradigms . . . . .	598
16.2 Case-Based Reasoning and Learning . . . . .	599
16.2.1 Components of a CBR System . . . . .	600
16.2.2 The Four "R"s of CBR . . . . .	600
16.2.3 Case Representation Structure and Determining Case Similarity . . . . .	601
16.2.4 Issues with CBR . . . . .	601
16.2.5 Case-Based Reasoning and the U.S. Legal System . . . . .	602
16.3 Learning through Description Generalization/Specialization . . . . .	602
16.3.1 General-to-Specific (G-S) and Specific-to-General (S-G) Approaches . . . . .	602
16.3.2 Representations and Descriptions . . . . .	603
16.3.3 The Training Sets . . . . .	603
16.3.4 Consistency and "Covering" . . . . .	603
16.3.5 Algorithmic Goals for Generalization/Specialization-Based Learning . . . . .	605
16.3.6 Description Modification . . . . .	605
16.3.7 Specific to General (S-G) ("Opening" the Description) Approach . . . . .	605
16.3.8 General-to-Specific (G-S) ("Narrowing" the Description) Approach . . . . .	606
16.3.9 Example: Learning 2-D Geometric Descriptors . . . . .	607
16.3.10 Inductive Reasoning and Truth/Falsity Preservation . . . . .	608
16.3.11 Generalization Operators . . . . .	609
16.4 Learning Decision Trees . . . . .	612

16.4.1	Synopsis of the ID3 Approach . . . . .	613
16.4.2	Sample Exemplars (ID3 Input Data) . . . . .	613
16.4.3	The Decision Tree: Definition and Tree Components . . . . .	613
16.4.4	The General Concept of (Information) Entropy . . . . .	614
16.4.5	The ID3 Algorithm . . . . .	615
16.4.6	ID3 Algorithm Examples . . . . .	618
16.4.7	An ID3 Application: Learning Musical Structures . . . . .	621
16.4.8	Another ID3 Handworked Example: Results and Ramifications . . . . .	625
16.4.9	Using ID3 for Learning Block Movement Strategies . . . . .	629
16.4.10	ID3 Learning, Training Data, and Generalization . . . . .	632
16.4.11	Converting a Decision Tree into a Rule Set . . . . .	640
16.5	Integrating and Implementing ID3, Generalization, and Rule Determination . . . . .	640
16.5.1	Exemplars . . . . .	640
16.5.2	ID3 Results . . . . .	640
16.5.3	Validation of the Tree . . . . .	641
16.5.4	Determination of Resulting Classification Rules . . . . .	642
16.6	Exercises . . . . .	642

## 17 Genetic Algorithms, Swarm Intelligence, and Other Evolutionary Computing

### Concepts in IS

649

17.1	Biology Suggests a Computation . . . . .	649
17.2	Genetic Algorithms . . . . .	649
17.2.1	Applications . . . . .	650
17.2.2	Chromosomes, Genes, the Genome, and Genotypes and Phenotypes . . . . .	650
17.2.3	GA as a Search and Optimization Technique . . . . .	651
17.2.4	Critical Aspects of a Genetic (Programming) Solution . . . . .	653
17.2.5	Genetic Algorithm Simulation Parameters . . . . .	654
17.3	A Simple Genetic Algorithm Example . . . . .	656
17.3.1	Problem Formulation . . . . .	656
17.3.2	Objective Function . . . . .	657
17.3.3	Selection, Crossover, and Mutation . . . . .	658
17.3.4	Sample Results . . . . .	659
17.3.5	Other Simulation Parameters and Examples . . . . .	670
17.3.6	Extension: Modifying the Fitness Function . . . . .	675
17.4	Applying Genetic Approaches to the Traveling Salesman Problem (TSP) . . . . .	679
17.4.1	The TSP . . . . .	680
17.4.2	Significance and Computational Complexity of the TSP . . . . .	680
17.4.3	TSP Representational Issues and Approaches . . . . .	681
17.5	Application of Genetic Algorithms to ANN Design and Training . . . . .	683
17.5.1	Topology Optimization and/or Weight Optimization . . . . .	684
17.5.2	The Blueprint Representation . . . . .	684
17.5.3	Blueprint-Based Crossover . . . . .	684
17.5.4	Sample Blueprint-Based Representation Results . . . . .	685

17.6 Fusing Learning and a Genetic Application: Meta-Algorithm Development . . . . .	685
17.6.1 Problem Motivation . . . . .	685
17.6.2 Background . . . . .	686
17.6.3 Dynamic Algorithm Graph Representation and Manipulation . . . . .	686
17.6.4 The Algorithm Graph Data Structure Representation . . . . .	687
17.6.5 Genetic Recombination of Graphs . . . . .	688
17.6.6 Crossover . . . . .	688
17.7 A Genetic Approach to Optimizing <code>gcc</code> Compiler Options . . . . .	691
17.8 Swarm Intelligence . . . . .	694
17.8.1 Natural Behavior . . . . .	694
17.8.2 Definition . . . . .	695
17.8.3 Population-Based Processing Approaches . . . . .	695
17.8.4 Ant Algorithms (How Real Ant Colonies Find a Minimum Path) . . . . .	695
17.8.5 Developing and Applying an Artificial Ant Colony Algorithm . . . . .	697
17.8.6 Return to the Traveling Salesman Problem . . . . .	697
17.8.7 Solving the TSP Using the Ant Colony Metaphor . . . . .	697
17.8.8 Computational Shortcomings of the Artificial Algorithm and the ACS . . . . .	698
17.9 References . . . . .	700
17.10 Exercises . . . . .	700

## A Fundamentals of Discrete Mathematics 705

A.1 Sets . . . . .	705
A.1.1 Basic Concepts . . . . .	705
A.1.2 Operations on Sets . . . . .	706
A.2 Relations . . . . .	706
A.2.1 Properties . . . . .	707
A.2.2 Relations As Functions . . . . .	707
A.3 Graphs . . . . .	708
A.3.1 Subgraphs . . . . .	708
A.3.2 Directed Graphs (Digraphs) . . . . .	708
A.3.3 Directed Graphs and Inset Relations . . . . .	708
A.3.4 Trees . . . . .	709

## B Fundamentals of Prolog 711

B.1 Introduction . . . . .	711
B.2 Predicates, Clauses, Facts, Rules, and Goals . . . . .	711
B.2.1 Variables . . . . .	712
B.2.2 Goals . . . . .	712
B.2.3 Examples . . . . .	713
B.2.4 The Anonymous Variable . . . . .	714
B.2.5 Prolog Uses Term Unification . . . . .	714
B.2.6 Prolog “help” Notation . . . . .	716
B.2.7 Recursion in Prolog . . . . .	716

B.2.8	Variable Bindings and <code>is</code> . . . . .	717
B.2.9	Prolog and Arithmetic . . . . .	717
B.2.10	Predicates to Verify the Type of a Term . . . . .	717
B.3	Lists in Prolog . . . . .	717
B.3.1	List Representation . . . . .	717
B.3.2	A List Example . . . . .	718
B.3.3	Prolog List Membership via Predicate “ <code>member</code> ” . . . . .	718
B.4	Prolog Databases . . . . .	718
B.4.1	Prolog Dynamic Databases . . . . .	718
B.4.2	<code>assert</code> and <code>retract</code> . . . . .	718
B.5	Backtracking and the cut (!) . . . . .	719
B.5.1	The Concept of Negation as Failure . . . . .	719
B.5.2	Logic in Prolog . . . . .	719
B.5.3	Prolog and Modus Ponens (MP) . . . . .	719
B.6	Parsing (Grammatical/Structural Recognition) and Prolog . . . . .	720
B.6.1	Using Grammar Rule Notation (the LGN) . . . . .	720
B.6.2	A Significant LGN Utility: Automatic Parsers . . . . .	721
B.6.3	Adding Variables in the LGN . . . . .	721
B.7	Prolog References and Pragmatics . . . . .	722
B.7.1	Books . . . . .	722
B.7.2	Web References . . . . .	722
C	<b>Fundamentals of Linear Algebra</b>	723
C.1	Elementary Matrices . . . . .	723
C.2	Vectors . . . . .	724
C.3	Linearity . . . . .	725
C.4	Inner and Outer Products and Applications . . . . .	726
C.5	Measures of Similarity in Vector Space . . . . .	727
C.6	Differentiation of Matrices and Vectors . . . . .	728
C.7	The Chain Rule . . . . .	729
C.8	Gradient Descent-Based Optimization Procedures . . . . .	729
D	<b>Fundamentals of Lisp</b>	733
D.1	The Roots of Lisp . . . . .	733
D.2	First Principles . . . . .	733
D.2.1	Programs and Data Are Lists . . . . .	733
D.2.2	The Lisp Top-Level Loop (EVAL) . . . . .	734
D.3	Common Lisp Building Blocks . . . . .	734
D.3.1	s-Expressions (or Forms) and Evaluation . . . . .	734
D.3.2	Some Special Forms . . . . .	735
D.3.3	Basic Function Groups . . . . .	735
D.3.4	Special Symbols . . . . .	736
D.4	References . . . . .	737

D.4.1	Implementations . . . . .	737
D.4.2	Online Common Lisp Resources and References . . . . .	738
<b>Bibliography</b>		<b>739</b>
<b>Index</b>		<b>755</b>



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

### 3. Developing IS Involves Software.

4. **Developing IS Requires Credentials.** In almost any endeavor, there are “talkers” and “doers.” The IS developer needs to be both, with an emphasis on the latter. Specifically, IS developers:

- ◆ Need to have or have access to application-specific expertise;
- ◆ Need to be cognizant of the myriad of available IS design approaches varying applicability, tradeoffs, and costs;
- ◆ Need to realize there probably exists more than one viable solution approach;
- ◆ Need to realize there probably exists more than one viable implementation;
- ◆ Need to realize that new problems probably have similarities with previously solved problems; and
- ◆ Probably need to use or develop some software.

## Audience

It is possible to use this text while deemphasizing implementation aspects. In the author’s opinion, however, the “hands on” or practical experience gained by developing and modifying working systems is an invaluable aspect of any IS course. Learning the jargon of IS is one thing; development of an ability to appreciate and experiment with the underlying concepts is a higher goal. Many of the ideas in IS are in fact only fully understood when the complexities and limitations of the problem present themselves to the student through hands-on experience.

The material presented herein is suitable for a first course in IS. This first exposure may occur anywhere from the junior level undergraduate to first year graduate level.

## History

In 1988, I wrote my first “AI” book.<sup>1</sup> At that time, AI emphasized a rule-based paradigm, implemented in Lisp or Prolog, with long discussions on uncertainty, learning, and search. Since that time, ontologies, the ubiquitous agents concept, fuzzy, neural, genetic, swarm, and other technologies have come into being. In addition, useful tools like Soar and CLIPS have become available. This book incorporates these concepts, and suggests that a single, simple, simplistic approach to IS is insufficient.

Intelligent systems research and development has fostered many “hidden” applications that often go unnoticed. This includes airline reservation systems, voice recognition, robotic toys for entertainment, and realistic animation. Intelligent, interactive, humanoid robots have yet to appear. In addition, many of the original and underlying problems are still with us. On the bright side, at least we no longer confuse intelligent systems success with chess-playing ability.

*It is impossible for a man who takes a survey of what is already known, not to see what an immensity in every branch of science yet remains to be discovered.*

Thomas Jefferson, Monticello, June 18, 1799

---

<sup>1</sup> Artificial Intelligence: An Engineering Approach, McGraw Hill.

## Acknowledgments

I would like to thank Tim Anderson at Jones and Bartlett Publishers for believing in this project. In addition, thanks are due to Amy Rose, Melissa Potter, and the Jones and Bartlett staff for producing a polished, readable text.

The trials of students at Clemson University who used draft versions of the text are appreciated.

Finally, the recognition of those who consider textbook production a bona fide and respectable academic venture is especially appreciated.

# Introduction to Intelligent Systems

## 1.1 What Is Intelligent Systems (IS)?

**Principles:** elementary truths, laws, or assumptions

**Paradigms:** a set of practices, often concerning an intellectual discipline

**Pragmatics:** practical considerations

**Intelligent system:** good question

**Intelligence through Computation.** Early computer pioneers realized almost immediately that the emerging computer had capabilities beyond number crunching [Neu58]. The Intelligent Systems (IS) field seeks to achieve *intelligent behavior through computation*. If useful and generally applicable “intelligent behavior” models and algorithms were available, this book would be much shorter and possibly superfluous. Therefore, this book considers a number of technical approaches aimed at developing autonomous *systems with intelligence*.

**What Is Intelligence?** *This is a surprisingly difficult question.* In fact, it has recently been cited as one of the five “deep” questions in computing [Win08]. For this reason, we defer to either the reader’s own definition or the myriad of definitions provided by dictionaries (see the problems), psychologists, philosophers, or even other IS researchers.

### 1.1.1 Key Topics or Themes

Any computation requires a representation of some entity (e.g., as a concept or as a numerical quantity) as well as a procedure for manipulation. Representation and manipulation (and learning) are key elements of IS. Obviously, we cannot manipulate something unless it is adequately represented.

Throughout this text, three terms (*representation*, *manipulation*, and *learning*) provide a common denominator for much of the study. The following are the three most relevant questions:

- How is the knowledge (or intelligence) represented?
- How is it manipulated?
- How is it learned (or acquired)?



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

**Summer Dartmouth Conference, 1956.** As noted previously, this conference may be the clearest indicator of the emergence of modern-day AI thrusts. As quoted from the proposal document:

*We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.*

Note the entire document is available online at <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html> as well as other sites.

Key areas investigated at the conference are now familiar topics and include:<sup>3</sup>

1. Automatic Computers
2. How Can a Computer be Programmed to Use a Language
3. Neuron Net
4. Theory of the Size of a Calculation
5. Self-improvement
6. Abstractions
7. Randomness and Creativity

### 1.2.2 Grand Research Challenges in Computer Science and Engineering Related to IS

With the leadership of the Computing Research Association and financial assistance from the National Science Foundation, a conference series on “Grand Research Challenges in Computer Science and Engineering” was initiated. In the Summer of 2002, a group of about 65 researchers from the public and private sectors convened to discuss the specific and urgent research challenges related to building computing systems of the future. The resulting document is available at <http://www.cra.org/Activities/grand.challenges/>

**The Five Overall Areas and Topics Related to IS.** The five areas noted in the Grand Challenges are:

1. **Create a Ubiquitous Safety.Net.** Providing a ubiquitous Safety.Net will save lives and minimize damage from disasters through timely prediction, prevention, mitigation, and response.
2. **Build a Team of Your Own.** Building cognitive partnerships of human beings with software agents and robots will enhance individual productivity and effectiveness.
3. **Provide a Teacher for Every Learner.** Tutoring each individual in a tailored, learner-centered format will enable people to more fully realize their potential.

<sup>3</sup>Topical headings are used.

4. **Build Systems You Can Count On.** Assuring reliable and secure systems from the regional electric grid to an individual's heart monitor will allow us to rely on information technology with confidence.
5. **Conquer System Complexity.** Building predictable and robust systems (with billions of parts) will enable broader and more powerful applications of information technology.

Many of the conclusions regarding technical objectives and direction in computing are related to computational intelligence. A few are summarized here:

- ◆ Data fusion and analysis: The ability to fuse related data from multiple sources to derive useful information is critical. Analysis and interpretation of collected data in conjunction with information databases (that could often be out of date and/or have partial or incorrect information) are essential.
- ◆ Ability to predict, prevent, mitigate, and respond to emergencies, coupled with research in augmented cognition: Augment human cognitive capabilities by developing machines capable of offloading human thought processes, yielding cognitive systems capable of actively supporting individuals in pursuing their goals and solving problems as they arise while planning for the future.
- ◆ Build a team of your own composed of robots and agents functioning as active partners: These systems enhance individual cognitive abilities by supplementing memory and problem-solving capabilities and by providing direct access to relevant data, expertise, guidance, and instruction. They work toward shared goals while understanding enough about the task, the individual, and each other to assist, mentor, cooperate, and monitor as needed.
- ◆ Incorporation of cognitive capabilities into both programs and machines, that is, to upgrade them from the status of automation tools to that of cognitive systems. Capabilities include:
  - Understanding tasks and tools as well as the context in which they are embedded
  - Acquiring knowledge by learning from experience and/or instruction
  - Representing their acquired knowledge in visual displays and in appropriate human and formal languages
  - Tracking the evolution of their environment over time and inferring critical consequences of what is known and sensed
  - Acting in a goal-oriented manner, while solving problems as they arise and planning for future contingencies
  - Operating over extended periods of time and across a broad range of domains
- ◆ Cognitive tutors: It has been demonstrated that it is possible for an automated tutor to improve student performance by roughly one standard deviation from the mean for some high school mathematics students. Significant human effort is required to develop the specialized knowledge; significant progress must be made in crafting knowledge representations that are both interoperable and reusable. The knowledge representations that underlie such tutors should also be designed to incorporate new knowledge about a subject area [Che05].

- ◆ Build systems with billions of parts: The limiting factor is software, not hardware. Every year, Moore's Law continues to pay its dividend, yielding ever higher raw performance per dollar. Software is still being written at the same manual pace and has progressed only incrementally since the 1950s. Software organization hits a "complexity barrier" somewhere above 10 million lines of code. Without better ways of producing and structuring software, the complexity barrier will constrain our ambitions for system behaviors.
- ◆ Building self-sustaining software today: Among the research challenges are:
  - Understanding and controlling emergent behavior
  - Learning in multi-agent systems
  - Negotiation and optimization
  - Architectures and networks
  - Programming languages and computational models

## 1.3 An Elusive Model Yields an Elusive Goal

Some aspects of IS concern **developing and relating mental models to real-world computation**. Surprisingly, we appear to know a great deal more about our physiological functioning (e.g., operation of the heart, immune systems, and nutrition) than the underlying operation of our conscious and subliminal mental processes.

### 1.3.1 The Mind and (Versus?) the Brain

The brain is a biological computer containing approximately 100 billion neurons (although the argument is often made that not a large percentage are used). In the brain, each neuron is connected to  $10^3$  to  $10^4$  other neurons. Note that as the number of neurons increases, the number of possible interconnections grows exponentially.

One IS design strategy could be based upon emulation of the intelligence exhibited by the human "mind." For purposes of argument, suppose we adopt the oft-cited mind-brain relationship: "the mind is what the brain does." In other words, the working brain is the hardware and the mind is the set of processes running on this hardware. Computationally, these processes might be assumed to be due to some large set of "software" running on the brain. In other words, the "mind" is the biological processing itself.

Cognitive neuroscientists attempt to identify and understand the thought processes of the mind. This includes memories, emotions, responses, and experiences, and how they are stored. Furthermore, cognitive neuroscientists attempt to unify this cognitive model with the underlying biological signal processing.

The IS design objective thus becomes describing these mental processes (in effect, reverse-engineering the software) in a manner that permits replication on nonbiological hardware with the same macroscopic behavior. In fact, functional MRIs (fMRIs) have been used for this purpose [Fri05].

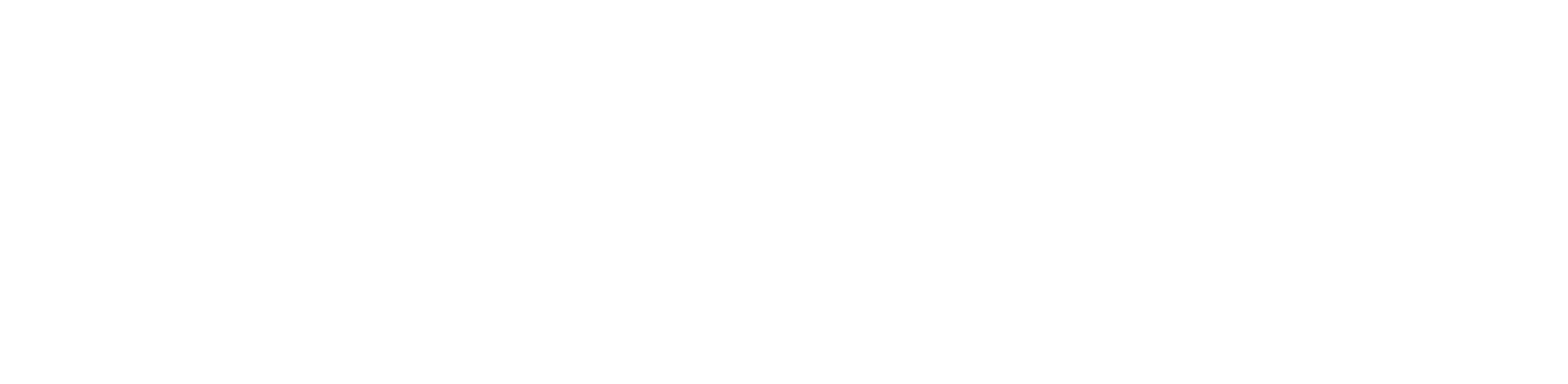
The difficulties with this approach are numerous. For one, as noted in Section 1.3, identifying the mental processes corresponding to the higher-level biological computation is somewhere between extremely difficult and impractical. In addition, the analogy itself is probably too simple. The brain and the mind have a somewhat circular coexistence; the mind can influence the brain and vice versa.



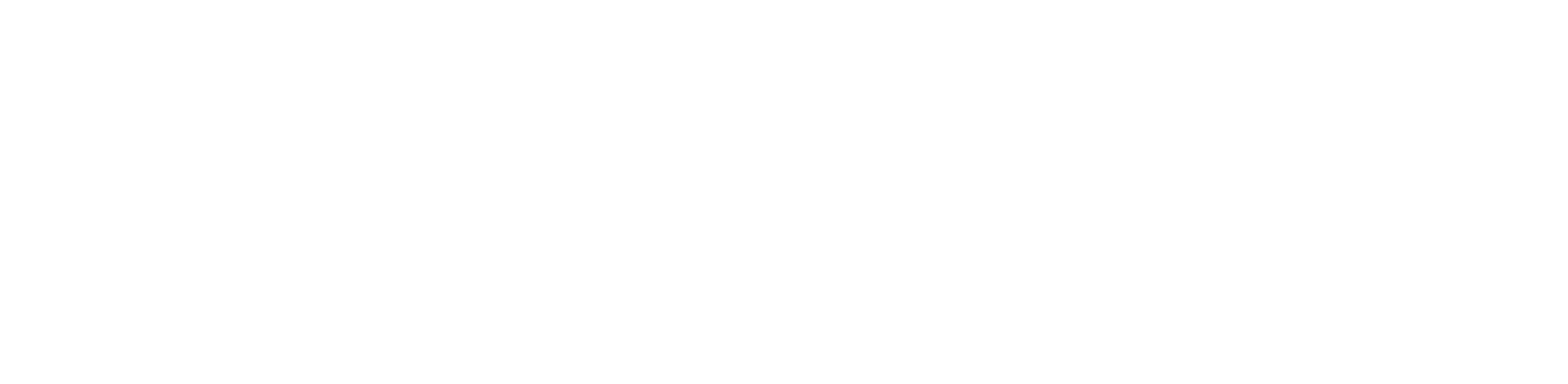
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

# First Steps in IS: Representation, Ontologies, and Obtaining Expertise

## 2.1 Representation in IS

### 2.1.1 Knowledge Representation and Related Definitions

A (knowledge) *representation* is a scheme or approach used to capture the essential elements of a problem domain. A *manipulable representation* is one that facilitates computation, and, for our purposes, is the only one worthy of further consideration. In manipulable representations, the information is accessible to other entities (e.g., inference engines), which use the representation as part of the overall computation.

Development of the representation may involve the process of *knowledge engineering*. This may include expert query. The issues involved in the development of a detailed knowledge representation are complex, interrelated, and problem and goal-dependent.

### 2.1.2 Human Reasoning: Classes, Categories, Concepts, and Chunking

In Chapter 1, Section 1.1.4, the role of psychology and cognitive science in IS was summarized. Modern psychology includes the study of the notions of categories (classes) and concepts. Categorization (or classification) is a basic capability of high-level biological systems [Ros99]. Whereas every object is unique, different objects may be conceptualized and/or treated similarly as members of a class or category. Aristotle argued that humans define categories by a set of characteristic features, in a formal, rigid definition of the category. Modern philosophy<sup>1</sup> (and pattern recognition) seems to refute this notion of category membership in two ways:

- Category (class) membership is not all-or-nothing (crisp), but rather fuzzy; and
- Category membership is determined by resemblance to one or more exemplars for the category. In other words, there is no single, all-encompassing set of features that define a category. Instead, a (possibly dynamic) set of category exemplars provide a reference point for membership in the category.

<sup>1</sup>Professor Eleanor Rosch has written extensively on the subject.

**Chunking.** Chunking is related to the study of memory theory, i.e., paradigms for how categories are defined, formed, and learned (or stored) in the human brain. For example, creating related “chunks” of information from experience is fundamental to human learning. New situations may be approached by recalling similar stored “chunks” identified in previous situations [GLC<sup>+</sup>01]. This aspect of cognitive science is not lost on IS development; the Soar system (Chapter 9) implements a form of this type of learning.

### 2.1.3 Declarative Versus Procedural Representations

Knowledge representation approaches may be subdivided into declarative (i.e., the representation of facts and assertions) and procedural (i.e., the storing of actions or consequences) representations. An alternative characterization of this representational dichotomy is knowing “what” (declarative) vs. knowing “how” (procedural).

The specifics of an application may dictate a preferred approach. A reasonable postulate is that versatile, intelligent systems may need to use both.

Declarative schemes include logic-based and relational approaches. Relational models may lead to representations in the form of trees, graphs, or semantic networks. Logical representation schemes include the use of propositional logic and, more importantly, predicate logic.

Procedural representational schemes store knowledge in how to do things. They may be characterized by formal grammars and usually implemented via procedural or rule-based (production) systems.

**Declarative Versus Procedural Knowledge: Examples.** The difference between declarative and procedural schemes may be illustrated by a simple example. A table of logarithms is an **explicit enumeration** of this (numerical) domain knowledge and would be considered a declarative representation. On the other hand, a stored sequence of actions indicating **how to compute** the logarithm of a number would be considered a procedural representation. This example illustrates tradeoffs in using declarative vs. procedural knowledge representations. Declarative representations are usually more expansive (and expensive), in the sense that enumeration may be redundant and inefficient. However, modification of declarative representations is usually quite easy; one merely adds or deletes the knowledge. Procedural representations, on the other hand, may be more compact, at the expense of flexibility.

## 2.2 Tools for Representation

### 2.2.1 Discrete Mathematics: Using Relations and Properties

**Binary Relations.** If  $A$  and  $B$  are sets, a binary relation from  $A$  to  $B$  is a subset of  $A \times B$ , where  $A \times B$  denotes the Cartesian product of the sets  $A$  and  $B$ . This definition is mathematically precise; it is referred to as a binary relation since it only involves two sets.

**Relation Properties.** A relation from set  $A$  to set  $B$  may be represented by a set of ordered pairs,  $\{(a_i, b_i)\}$ , where  $a_i \in A$  and  $b_i \in B$ . There are three properties of prime importance for a relation  $R$ :

1. reflexive:  $R$  is reflexive if  $\forall a \in A, (a, a) \in R$

2. symmetric:  $R$  is symmetric if  $\forall(a, b) \in R, (b, a) \in R$
3. transitive:  $R$  is transitive if  $\forall(a, b) \in R$  and  $(b, c) \in R, (a, c) \in R$

**Using Relations in IS Representations.** Suppose our knowledge representation involves a relation between two entities,  $x$  and  $y$ .  $x$  and  $y$  may be symbols, numbers, actions, or concepts. Consider, for example,

“attribute  $R$  of object  $x$  has value  $y$ ”

For representation purposes, one view might be that  $x$  and  $R$  are arguments to a function, denoted  $f$ , which returns  $y$ , i.e.,  $y = f(x, R)$ . This relation may also be viewed as the ordered triple  $(R, x, y)$ , which is in the (attribute, object, value) form or  $(A, O, V)$  format. This is read as “the  $A$  of  $O$  is  $V$ .” An alternative is to view this mapping as a relation, denoted  $R$ , (in the strict mathematical sense), i.e.,

$x$  is related to  $y$  by relation  $R$

or  $R = \{(x, y)\}$  or graphically  $x \xrightarrow{R} y$ .

## 2.2.2 Digraphs to Semantic Nets

The use of graphical constructs (directed graphs or digraphs) to enumerate both numerical and symbolic relations among sets of entities is common.<sup>2</sup> A semantic network, or simply semantic net, is a labeled digraph used to describe relations (including properties) of objects, concepts, situations, or actions. Semantic nets are ideal visualization tools that are widely used in IS to facilitate knowledge representation. Knowledge in a semantic net may be naturally organized to reflect hierarchies and thus enable inheritance.

Unfortunately, for most reasonable representations of our knowledge about even a very restricted domain, the overall semantic net is usually a large and highly interconnected entity.

## 2.2.3 Semantic Net Examples

A semantic net represents objects as nodes (shown as circles) and relations as labeled arcs (or edges). For “real-world” entities, they are usually quite complicated. A relatively simple semantic net for a “blocks world” situation is shown in Figure 2.1. Figure 2.2 shows a portion of a semantic net for a typical vehicle description.

Observe from these figures that:

1. The semantic net consists of many different edge labels or relations, such as “is-a,” “has-attribute,” “used-for,” “adjacent-to,” etc.
2. The semantic network may contain redundant or “derivable” information. For example, the “left\_of” and “right\_of” arcs between two nodes shown in Figure 2.1, while correct, are unnecessary, since given one, the other may be deduced. In addition, the enumeration of other

---

<sup>2</sup>In fact, as we show later, ontology editors may produce these graphical constructs, or even allow the development of ontologies graphically.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

Protege also supports development using the OWL. The OWL language incorporates formal semantics that allow the derivation of logical consequences, i.e., new facts not explicitly present in the ontology. Defining a class with attributes using OWL is not as direct as the frame-based approach. The OWL and frame-based distinctions are addressed in the problems. Note that a “full” installation of Protege (3.4 or later) supports both formats.

We introduce simple Protege examples and some underlying implementation issues in this Chapter. Many other features and examples of Protege use are included in Chapter 8.

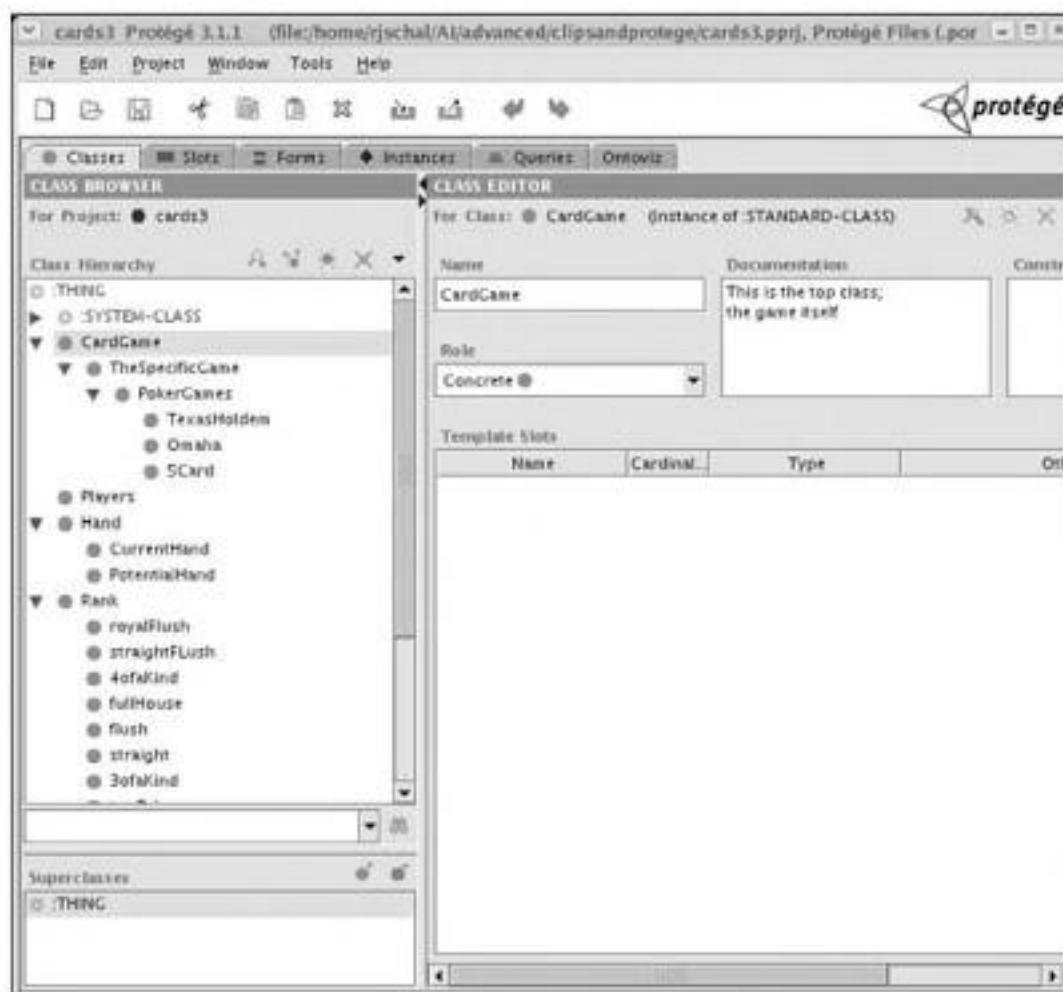
## 2.6.1 Protege Examples

### Card Games

Figure 2.8 shows the use of Protege in defining an ontology for a set of card games. A graphical representation of the resulting class hierarchies is shown in Figure 2.9. One of the attractive features of Protege is the ability (using so-called plugins) to graphically display and input elements of an ontology.

### A More Complex Protege Ontology Example

Consider the ontology view shown in Figure 2.10. This ontology was inspired by the hierarchical representation of Figure 2.3. We reuse this ontology in Chapter 8 to show integration with CLIPS COOL. A graphical representation of the ontology, including instances, is shown in Figure 2.11.



**Figure 2.8** An (Incomplete) Ontology for Card Games Produced Using the Protege Ontology Editor

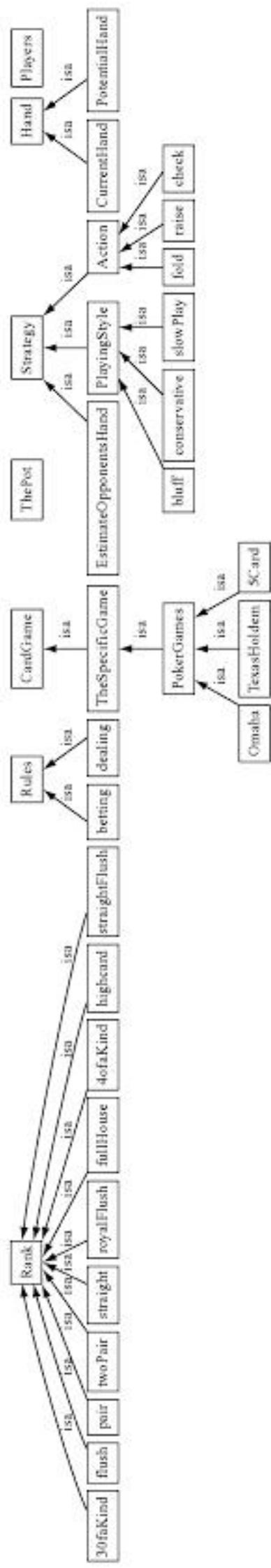


Figure 2.9 Graphical Representation for the Card Games Ontology (Produced Using Protege)

## 2.6.2 Alternate Representation Formats Using Protege

Our primary interest is in using the Protege pins and pont representational formats (incorporated in files ending with these extensions). This allows direct importation of the Protege-based knowledge representation into CLIPS. One of the more noteworthy aspects of Protege is the ability to import and export a number of standardized ontology representations, including CLIPS, OWL, RDF, XMK, and HTML. For example, the HTML representation (shown displayed on a browser) for the Protege-based ontology shown of Figures 2.10 and 2.11 is shown in Figures 2.12 and 2.13.

# 2.7 Ontological Reasoning and "Common Sense" Implementations

## 2.7.1 CYC (OpenCYC)

The objectives of CYC Corporation's CYC effort is "Formalized Common Knowledge." OpenCyc is the open source version of what is claimed to be the world's largest and most complete general knowledge base and commonsense reasoning engine. It may be used as the basis of a wide variety of IS applications. Versions for both Linux and Windows XP operating systems are available.

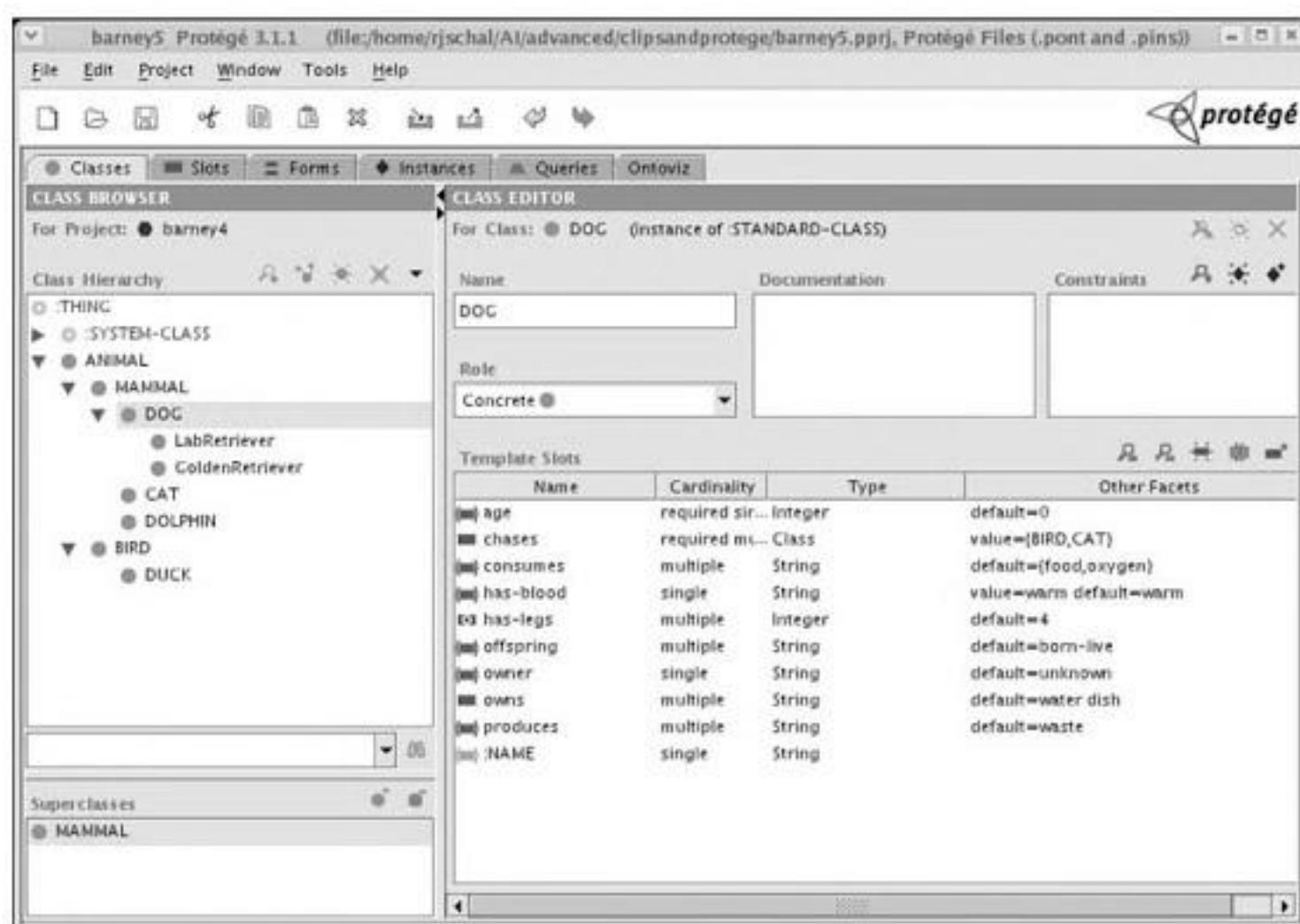


Figure 2.10 Protege Structured Representation (Class View Shown) Example



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



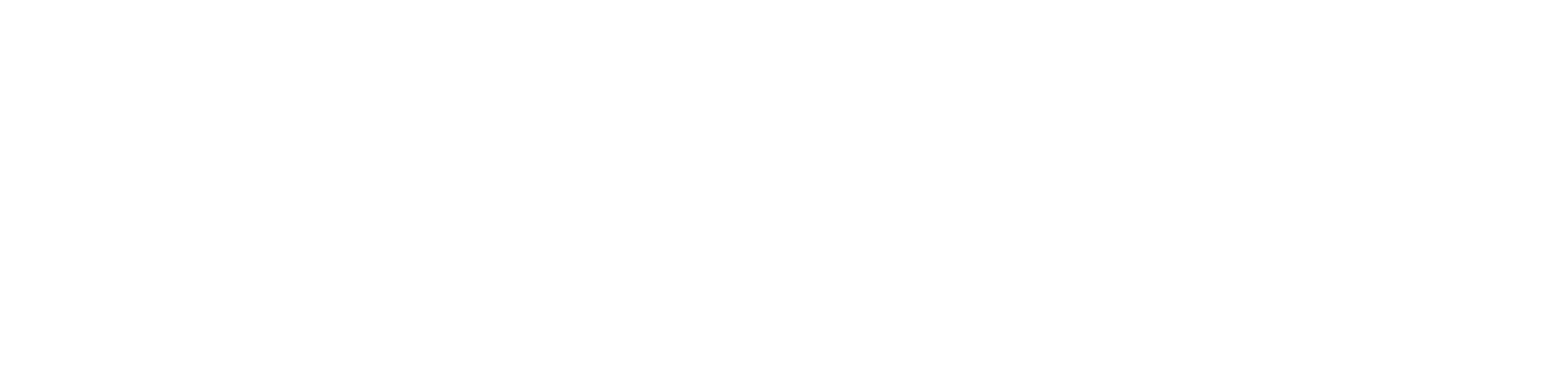
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

9. Consider an alternative structured representation based upon the *typical* attributes of an entity. For example, instead of representing “all elephants are gray” or equivalently

$$(\forall X) \text{ elephant}(X) \rightarrow \text{gray}(X)$$

we choose to represent

“*typical* elephants are gray.”

Can this approach be quantified and extended? How are exceptions handled?

10. Using Protege (and good judgment), develop an ontology for the concept: *Computer Engineering and Science (undergraduate college-level) Course*. Based upon this, extend your ontology with instances to realize a knowledge base. Show (using the graphical features and associated plugins in Protege) graphical depictions of the relations between elements of your ontology.

Be sure to develop answers to the basic questions posed in Section 2.5.2, and develop competency questions.

11. Would you consider an ontology produced by Protege to be the basis for a declarative or a procedural knowledge representation? Explain.

12. Using Protege, develop an ontology for a world consisting of two blocks, **Block1** and **Block2**, as shown in Chapter 9, Figure 9.22. Both blocks are situated on an  $n \times n$  checkerboard at initially different locations. Each block may sense its location as well as the location of the other block. In addition, each block may move into an empty adjacent cell. The objective (motivation) of **Block2** is to move toward and overtake **Block1** (get to the same location). **Block1** has a very different objective (motivation)—that is, to not be overtaken or caught.

13. Using Protege, develop an ontology for the domain of programming languages. Keep in mind that they may differ in terms of paradigm, language constructs, intended application, ease of use/learning, etc.

14. Section 2.1.3 considered declarative vs. procedural representations. In this context, is a **Makefile** (used in conjunction with the **make** utility) a declarative or a procedural representation?

15. Must a semantic net based solely upon an “is-a” link or relation yield an acyclic, hierarchical graph?

16. The fact that all dogs are mammals may be represented in logic by the statement

$$(\forall X) \text{ dog}(X) \rightarrow \text{mammal}(X)$$

How would you represent this in Prolog?

17. More general logical representations, which relate class values, are possible. For example, if the class of entity  $X$  is  $t1$ , then the value of property  $p1$  (of  $X$ ) is  $v1$  may be written:

$$(\forall X) \text{ class}(X, t1) \rightarrow p1(X, v1)$$

Convert this to a Prolog representation.

18. Suppose you are developing an ontology in Protege. How would you determine whether the OWL or the frames-based representation is most appropriate for your ontology?
19. We have noted that realistic ontologies tend to be very large (for example, in terms of number of frames and instances). As a practical matter, is there a limit to the size of an ontology developed in Protege?
20. Develop a Protege ontology to be used for temporal reasoning. Elements should include concepts such as time, measures of time (years, months, days, etc.), and instances such as January, Tuesday, etc.
21. Using Protege, develop an ontological representation of the semantic net shown in Figure 2.2.
22. Is there a difference between a knowledge representation and a knowledge base?
23. One definition sometimes proposed for an ontology (Section 2.5.2) is “*a taxonomy with multiple link types, each with precise meaning.*” Is this definition accurate?

# Search and Computational Complexity in IS

## 3.1 Why Do We Care about IS Problem Computational Complexity?

The concepts of computational complexity and search have a strong link to the practical aspects of IS development. An algorithm used to solve an IS problem is of little use if it cannot provide a solution, for a reasonable problem size, in reasonable time, and using reasonable computing resources. The term “reasonable” is used here to denote practical or problem-dependent limits. For example, in processing television images, it is necessary (in the United States) to complete processing of a frame of data in  $\frac{1}{30}$  sec., regardless of the image resolution. Moreover, if the problem scale changes (e.g., the resolution of a digital image doubles), it is important to be able to predict the required increase in the computational time required for processing (or the additional computational resources necessary to achieve the processing result in the same time).

We often focus on the complexity of an algorithm in terms of required computations as a function of the problem size. However, computational resources (as a function of problem size) such as the *space bounds* of the computation also warrant consideration. For example, space bounds could include the amount of available memory required (without swapping). Before exploring the concepts of computational complexity and search, we digress to introduce the notion of an IS *state space*.

## 3.2 The Concept of a System State Space

Readers familiar with linear algebra may have been introduced to the concept of a vector space. A similar concept that is extremely important in IS is the notion of a system *state space*.

### 3.2.1 System State Representation

The system state, or state description, contains the set of all information, in the context of the chosen representation, which describes the current status of the system. The system state is a somewhat abstract concept until we consider specific examples of state representations. States  $S_i$  and  $S_g$  denote the initial and goal states, respectively, of a system.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

a position (node) in the graph. Here we have assumed specification of initial and goal states. As we shall see, many search algorithms are based upon a performance measure (estimator of distance between initial and goal states).

### 3.4.3 Definition of a Search Problem ( $P$ )

A search problem,  $P$ , is characterized by a 4-tuple:

$$P = \{D, S_i, T, G\}$$

where

$D$  is a set of system states that represent the problem state-space;

$S_i \in D$  is a starting state;

$T = \{t_1, t_2, \dots\}$  is a set of transformations (e.g., operators, rules);

and

$G \subset D$  is a set of goal states.

A solution to  $D$ , denoted  $T_s$ , is a sequence of  $t_i = t_1, t_2, \dots, t_p$  with the property that

$$t_n(t_2(t_1(S_i))) \in G$$

Note that  $D$  and  $T$  may be finite or countably infinite. We may partition  $D$  into two mutually exclusive subsets:

1. Those  $D_r \subset D$  where  $\exists a T_s$  such that

$$t_n \cdots (t_2(t_1(S_i))) \in D_r$$

These are so-called *reachable states*.

2. Those  $D_{ur} \subset D$  for which no  $T_s$  exists such that

$$t_n \cdots (t_2(t_1(S_i))) \in D_{ur}$$

These are *unreachable states*.

If

$$G \cap D_r \neq \emptyset$$

then  $P$  is solvable. Unfortunately, it is difficult, at best, to ascertain that  $P$  is solvable before attempting a solution.

### 3.4.4 State-Space Graphs

A state-space graph (SSG) is a digraph that enumerates  $D$  and  $T$ . Specifically, nodes of the SSG represent elements of  $D$ , and directed arcs represent elements of  $T$ , i.e., each  $t_i$ .  $T$  generates an arc in the SSG between nodes  $d_i$  and  $d_j \in D$  to indicate  $d_j = t_i(d_i)$ .

### 3.4.5 Search Costs and Heuristics

Using the SSG, a search cost function,  $C$ , is a mapping of all arcs onto the set of non-negative real numbers. More intuitively,  $C$  is a mechanism to associate a cost measure with a path through the SSG. Typically,  $C$  is used along with a search heuristic/algorithm to identify paths with low cost.

### 3.4.6 Graphical Representations for Search

We take a more detailed look at the graphical representations that may be employed in IS problems. Note that the principal utility of this abstraction is in conveying the complexity of the solution process, and in comparing alternative solution approaches. Once a problem representation has been chosen, the problem space, within which the search process occurs, may be explored. In order to develop a unified approach that is applicable to a number of IS problems, often graphical abstractions are chosen. Particular states of the system are represented by nodes in a digraph, and the edges in the graph, that link nodes (or states) represent the application of a state modification entity (i.e., a rule or operator). Note that this graphical representation does not assume certain system properties, such as decomposability or commutativity. If these properties exist, however, modifications, partitioning, or simplifications of the graph are possible. Thus, in large-scale problems, this graph is both large and complex.

Nodes are interpreted as states. A state might be the current contents of a system database. Actions (rule firings or planning operator applications) on all possible states, describe a graphical construct showing a linked sequence of system states. In most problems, one such state, denoted  $S_i$ , represents the system initial state, and one (or more) denote the goal state,  $S_g$ .

### 3.4.7 Example: A Graphical Representation of State Propagation

Consider the following simplistic rule-based system:

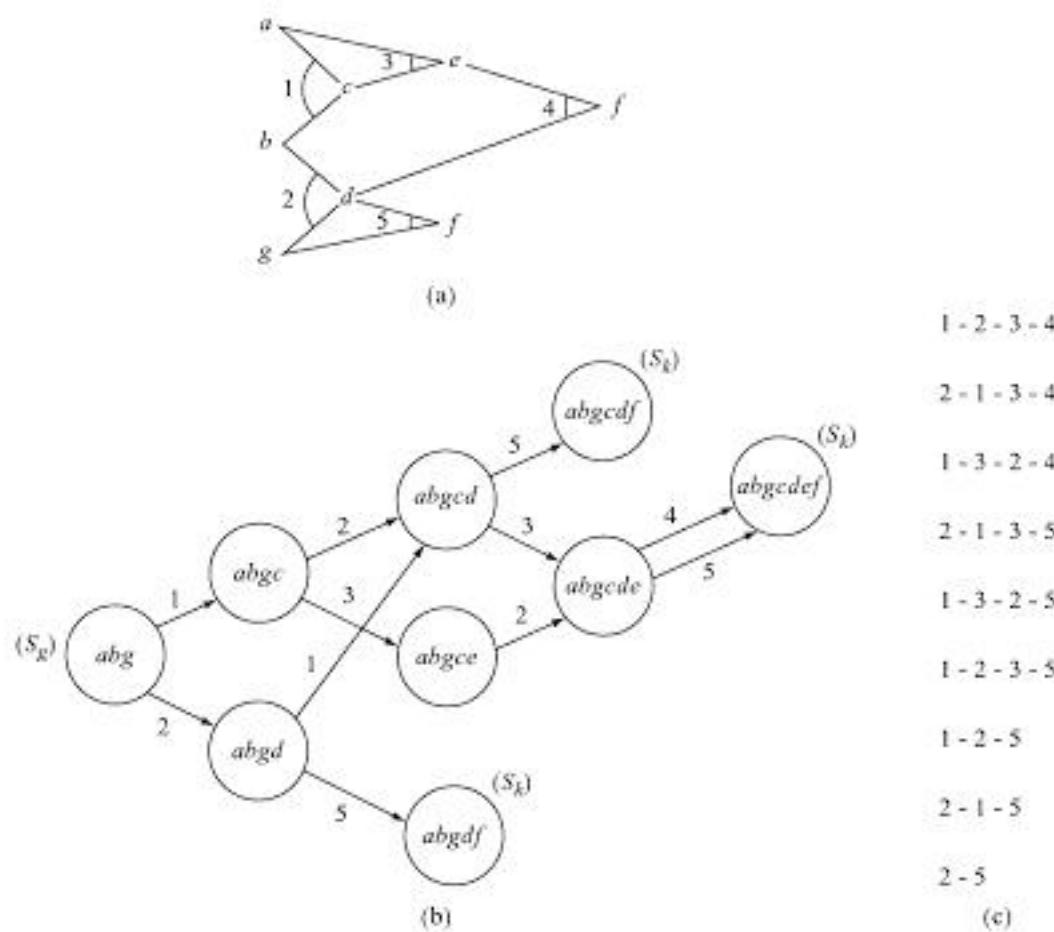
Initial (database) facts:  $a \ b \ g = S_i$ .

System operators (rules) are shown as follows:<sup>2</sup>

Rule 1	IF	a	b	THEN	c
2	'	b	g	'	d
3	'	a	c	'	e
4	'	e	d	'	f
5	'	d	g	'	f

Suppose the objective is to produce a state that contains fact **f**. Figure 3.8 (a) shows a combined inference net for the system. More importantly, Part (b) shows the search space and possible paths that lead from  $S_i$  to (one)  $S_g$ . Note the nine possible paths (rule-firing sequences) are shown in Table 3.1.

<sup>2</sup>Note "THEN" means "Add the fact."



**Figure 3.8** Problem Search Space. (a) Inference Net for Rules. (b) Search Paths Through State Space Using Forward Chaining. (c) Possible Rule-Firing Sequences.

1 - 2 - 3 - 4

2 - 1 - 3 - 4

1 - 3 - 2 - 4

2 - 1 - 3 - 5

1 - 3 - 2 - 5

1 - 2 - 3 - 5

1 - 2 - 5

2 - 1 - 5

2 - 5

**Table 3.1** Possible Rule-Firing Sequences Leading to Goal States.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

### Breadth-First ("Vertical") Search

A simple problem search space is shown in Figure 3.18(a). The breadth-first search path through this space is illustrated in Figure 3.18(b)(ii). Note that breadth-first search generates all nodes in the problem space tree at level  $i$  prior to exploration of the nodes at level  $i + 1$ . The computational complexity of any search algorithm is a function of the number of nodes investigated. The complexity of breadth-first search up to level  $q$ , therefore, is

$$C_{\text{breadth-first}} = 1 + b + \dots + b^q$$

where  $b$  is the branching factor. The complexity of this approach is exponential, since  $e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots$ .

Notice that since breadth-first search examines all nodes in the problem space tree at level  $i$  prior to investigating level  $i + 1$ , the path of shortest length is found. If path length is the metric used to measure (or compare) search algorithm performance, then breadth-first search is optimal. However, if all the goal states are a large distance from the start node and the number of alternative paths to store is large, breadth-first search may be impractical due to both time and memory requirements.

The general rule is that if the search has  $b$  branches at each node and goes to depth  $d$ , then there are approximately  $b^d$  nodes to explore. In these path/state problems, the branching factor  $b$  corresponds to the number of states adjacent to a given state, and the depth  $d$  corresponds to the path length, which is bounded by the number of legal states if the search is restricted to simple paths. For example, there are  $2^{58}$  nodes in a balanced binary tree of depth  $d = 58$ .

### Depth-First ("Horizontal") Search

Depth-first search explores all paths in a branch before considering alternative branches. It may be shown that the depth-first approach to level  $d$  generates the same number of nodes as in the breadth-first search (in a different order, however), and therefore has the same complexity. One interesting characteristic, however, is that it only requires storage of the current path, which is a function of the depth,  $d$ . For this reason, depth-first search is often preferred to breadth-first search. Figure 3.18(b)(i) illustrates the path or sequence and compares it with the breadth-first search.

### Forward-Backward or Bidirectional Search

This strategy involves starting both at  $S_i$  and  $S_g$  and working forward from  $S_i$  and backward from  $S_g$ , until a common state,  $S_c$  or states that "link"  $S_i$  and  $S_g$  is found. Thus, two breadth-first searches are required, although it is hoped that the combinatorial explosion in each remains limited before  $S_c$  is found. This is illustrated by merging Figures 3.16 and 3.17.

## 3.10 Other Search Approaches (Informed Search)

Many reasonable combinations and modifications of the preceding algorithm are possible. For example, depth-first search could be combined with breadth-first search in a number of ways. One example of this is depth-first with iterative deepening. An alternative is the incorporation of one or more heuristics.

## The Means-End Approach

Means-end is a valuable heuristic that may be employed in a variety of ways. The basic approach is to select operators that seem to provide a means to some end, most often the goal. For example, an often problem-specific and heuristic measure  $D(S_c, S_g)$  is used, at each step, to choose an operator that makes the decrease in  $D(S_c, S_g)$  largest.

### Best-First

Best-first search considers all the nodes that have been examined up to the current time, and proceeds by expanding the one that, *according to a heuristic*, is closest to the goal. In this sense, the best-first approach does not explicitly consider the cost of getting to node  $n$ , but rather only the estimated remaining cost. Intuitively, the search proceeds by “following the best guess.” Thus, best-first search may proceed in depth-first fashion. Again, a function or heuristic to compare two transformations (or their resulting states) is necessary.

## The Branch-and-Bound Search Approach

Practical solutions to many problems may be obtained by using a breadth-first (or best-first) search with some form of heuristically-guided pruning. A search technique known as branch-and-bound (B&B) is often applied to discrete and combinatorial optimization problems and games. The solution is found by elaborating on the best set of paths found so far. Clearly, B&B requires a measure of goodness, so that “best” may be quantified. Unfortunately, general goodness measures are often difficult to find.

B&B starts in a feasible region of the state space, called the root problem, and systematically enumerates a subset of the problem state space by considering succeeding states. Each solution in the feasible region is attributed with a measure of goodness. Assume it is desired to minimize this measure (e.g., often path length is used for illustration). The feasible region is subdivided into some number of subregions. The B&B search algorithm is applied recursively to these subregions. If a solution is found in a subregion, it is a feasible (but not necessarily optimal) solution to the full problem. The goodness measure attached to this feasible solution is then used to prune the remainder of the search tree by examining the costs associated with unexpanded nodes in the feasible region. If the lower bound for a node exceeds the best known feasible solution goodness measure, the node and any subsequent subtrees can be removed from consideration. The B&B search proceeds until all nodes have been explored or pruned.

## The A\* Algorithm

The A\* algorithm is a combination of the best-first and B&B procedures. It is based upon formulating a figure of merit (again often heuristic) for node  $n$  by tallying the actual (minimal or optimal) cost of the path from the initial state to node  $n$  (denoted  $g(n)$ ) and the *inferred or estimated* (minimal) cost from node  $n$  to the goal, denoted  $h(n)$ . If there is no path from node  $n$  to the goal,  $h(n)$  is infinite. Similarly, if no path between the initial state and node  $n$  exists, then  $g(n)$  is infinite. Therefore, the overall cost metric at node  $n$  is:

$$f(n) = g(n) + h(n)$$

which represents the cost of the optimal path that includes node  $n$ . In terms of traditional optimization problems,  $g(n)$  is the “cost to date” and  $h(n)$  is the “cost to go.” The objective is to find the optimum

path, i.e., one that yields a minimum  $f$  and links  $S_i$  and  $S_g$ . One difficulty, and thus limitation of this approach, is the determination of a good estimator for the quantity  $h(n)$ .

## 3.11 Exercises

1. Referring to the “checkerboard” problem representation of Section 3.3.2, consider the four operators whose effect upon the system state space is somewhat obvious:

- (a)  $move - up - x$ ,
- (b)  $move - down - x$ ,
- (c)  $move - left - x$ , and
- (d)  $move - right - x$

where  $x$  is the name of a block, i.e.  $x \in \{A, B\}$  in Figure 3.6. Characterize each operator as a relation on the system state, both as a set and graphically.

2. Develop a three-dimensional plot of the number of distinct states for the “checkerboard” problem of Section 3.3.2 as a function of  $n$  and  $q$  for the case  $p = 0$ . Hint: use a logarithmic scale for  $|S|$ .
3. The game of Go is played on a rectangular board of any size, but a board with 19 lines by 19 lines is commonly used. Surprisingly, Go is an old game; it originated in China over 3000 years ago. Worldwide, Go is and has been one of the most played games.

In game theory terminology, Go is a zero-sum, deterministic strategy board game, thus putting it in the same class as chess and checkers. Although the rules of Go are simple, the skill required to become a good Go player (or to develop good Go-playing software strategies) is significant. Whereas current computer programs are capable of defeating skilled human chess players, a mediocre Go player may easily defeat the current crop of Go-playing programs.

- (a) Look up and become familiar with the rules for playing Go.
  - (b) Determine (look up) the computational complexity of Go. How does it compare to chess?
  - (c) Explore (human) strategies for playing Go.
  - (d) Consider the implementation of a Go-playing program using some form of state prediction and search.
  - (e) At <http://www.gnu.org/software/gnugo/> you will find GNU Go, a freely distributed program that plays the game of Go. Download, build, and experiment with the program. What is the playing strategy implemented in GNU Go?
4. Estimate the computational complexity of the following problems, applications, or algorithms (and cite any assumptions used):
- (a) The board game of checkers
  - (b) Processing an  $n \times n$  image on a pixel-by-pixel basis
  - (c) Implementation of a multilayer, feedforward ANN (see Chapter 13)



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

## Prolog Results

```
?- goal1.
R1=car R2=road R3=grass R4=trees R5=road R6=grass
R1=car R2=road R3=grass R4=trees R5=sky R6=grass
R1=car R2=road R3=grass R4=trees R5=grass R6=grass
R1=car R2=road R3=trees R4=grass R5=trees R6=trees
R1=car R2=road R3=trees R4=grass R5=road R6=trees
R1=road R2=grass R3=trees R4=road R5=car R6=trees
R1=road R2=trees R3=grass R4=road R5=car R6=grass
R1=sky R2=trees R3=grass R4=road R5=car R6=grass
R1=grass R2=trees R3=grass R4=road R5=car R6=grass
R1=trees R2=grass R3=trees R4=road R5=car R6=trees

No
?- goal2.
R1=trees R2=road R3=trees R4=grass R5=trees R6=trees
R1=trees R2=grass R3=trees R4=road R5=trees R6=trees

No
?- goal3.

No
?-
```

## 4.7 The Map Coloring Problem

### 4.7.1 Background and History

The map coloring problem is another popular CSP example. It is very similar to labeling. The objective is to color the regions in a given map using a predefined number of colors such that adjacent regions (i.e., sharing a border) have different colors. This problem has a rich history and raises numerous theoretical issues.

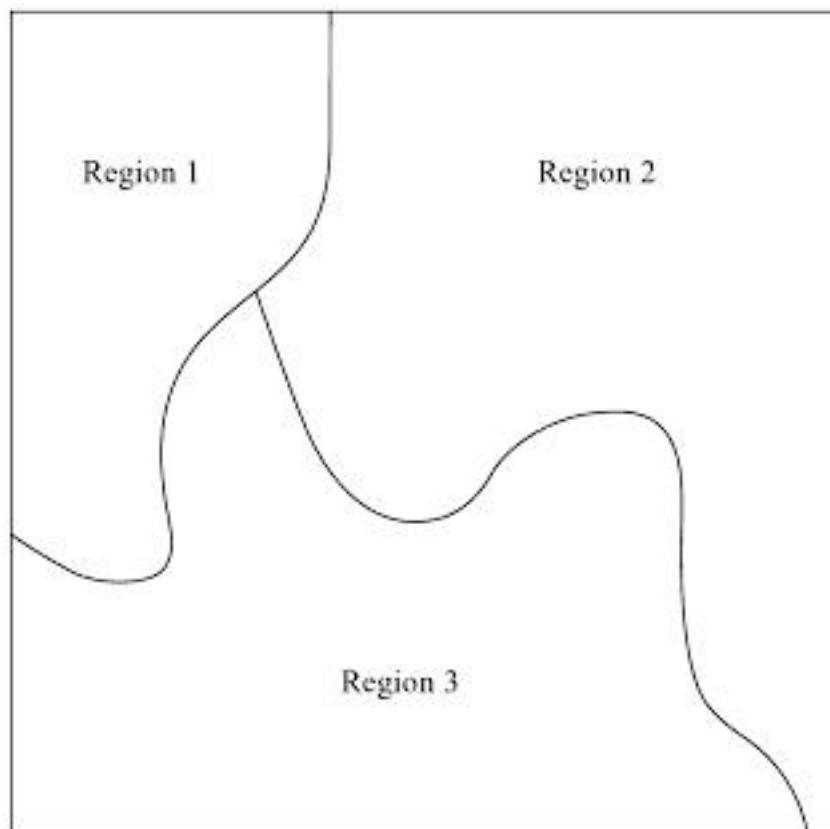
The problem may be approached via graph theory. In this interpretation, the graph vertex coloring problem is a CSP with the objective to color the vertices of a graph using a predefined number of colors so that the vertices that are connected by an edge have different colors.

The map or graph coloring problem belongs to a class of problems that are called NP-hard problems, for which no efficient solution method is known. The TSP is another example.

In the mid-1800s this problem was popular within a group of mathematicians since it was a challenging and unsolved problem. It was common knowledge among practicing mapmakers, even at that time, that four colors were sufficient to color a map. This wasn't actually proved, however, until 1976.

Furthermore, as we explore later, even though four colors are enough to color any map, some maps can be colored with fewer than four colors. In this problem we consider the use of three colors as well as four, and then extending the formulation to minimize the actual number of colors used. Note that a method to determine exactly how many colors are needed for a given map is itself a challenging problem.

The so-called four-color theorem states that any map in a plane can be colored using four colors in such a way that regions sharing a common boundary (other than a single point) do not have the same color. F. Guthrie first conjectured the theorem in 1853 and communicated it to the famous deMorgan.

**Figure 4.8** Map to be Colored

In 1878, Cayley wrote the first paper on this conjecture. A series of fallacious proofs followed. The conjecture that four colors were sufficient was finally proved by Appel and Haken in 1977 [AH77] in a computer-assisted proof.

### 4.7.2 Prolog-Based Solution Approaches

Refer to the map to be colored in Figure 4.8. We will use this simple map to develop Prolog-based map coloring CSP results.

**Prolog Formulation.** A Prolog formulation for the map of Figure 4.8, using three colors (abbreviated as “r,” “g,” and “b”) is shown as follows:

```
/* mapcolor1.pro */
/* first map case (3 regions) */
/* 3 colors; r,g,b */
/* write and fail used to print all solutions */

adj-to(r,g).
adj-to(r,b).
adj-to(b,g).

/* symmetry of adjacency relation */

adjacent-to(R1,R2) :- adj-to(R1,R2).
adjacent-to(R1,R2) :- adj-to(R2,R1).

goal :- adjacent-to(R1,R2),
       adjacent-to(R2,R3),
       adjacent-to(R1,R3),
```

```

write('Solution is'),nl,nl,
write('region 1 is colored: '),write(R1),nl,
write('region 2 is colored: '),write(R2),nl,
write('region 3 is colored: '),write(R3),nl,
fail.
```

Results using this formulation are shown here.

```

?- ['mapcolor1.pro'].
['mapcolor1.pro'].
% mapcolor1.pro compiled 0.00 sec, 1,888 bytes

Yes
?- goal.
Solution is

region 1 is colored: r
region 2 is colored: g
region 3 is colored: b
Solution is

region 1 is colored: r
region 2 is colored: b
region 3 is colored: g
Solution is

region 1 is colored: b
region 2 is colored: g
region 3 is colored: r
Solution is

region 1 is colored: g
region 2 is colored: r
region 3 is colored: b
Solution is

region 1 is colored: b
region 2 is colored: r
region 3 is colored: g
Solution is

region 1 is colored: g
region 2 is colored: b
region 3 is colored: r

No
?-
```

This effort is continued in the exercises, using different region data and a varying number of colors.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

$$\begin{aligned}\Gamma_1 &= \{\text{sky}\} \\ \Gamma_2 &= \{\text{trees}\} \\ \Gamma_3 &= \{\text{road, grass}\} \\ \Gamma_4 &= \{\text{road, grass}\} \\ \Gamma_5 &= \{\text{trees, road, car, grass}\} \\ \Gamma_6 &= \{\text{road, grass}\}\end{aligned}$$
**Figure 4.14** Limit Set Using Constraint  $\Gamma_1 = \{\text{sky}\}$  and  $\Gamma_j = \Gamma$   $j \neq 1$ 

**Results.** Notice from the performance results in Figure 4.16 that the unmodified Prolog solution required 219,842 subgoal call and redo invocations. The limit set-based version, however, required 7694 subgoal call and redo invocations. In other words, the limit set-based solution required only  $7694/219,842$  or 3.5% of the search required by the unmodified solution. Thus, the example provides a clear sign that restricting the possible label sets by preprocessing had a positive influence on the run-time behavior and efficiency of the solution.

In addition, notice the preceding limit set (the single unary constraint) produces 32 6-tuples. These would need to be verified to identify possible solutions in the limit set-modified Prolog implementation. By comparison, the single unary constraint requires the highest region to be labeled “sky”; thus there are  $5^5 = 3125$  possible tuples to be explored in the unmodified Prolog approach. This produces a ratio of search-related computation of  $\frac{32}{3125} = 0.01$ , i.e., the limit set-based solution should require only 1% of the brute-force search solution.

### Propagating $\underline{\Omega}_i$

While the handworked solutions using LRR/LDR are useful for illustration of the iterative solution process, a more automated procedure is desired. While developing the notation for the overall problem in Section 4.8.1, it was noted that vector  $\underline{\Omega}_i$  (corresponding to current label set  $\Gamma_i$  on  $u_i$ ) may be iteratively propagated. Once unary constraints are incorporated into the label sets through  $\underline{\Omega}_i^1, \underline{\Omega}_i^k$  may be propagated as follows:

$$\underline{\Omega}_i^{k+1} = \otimes|_{i \neq j} (\Omega_{ij} \underline{\Omega}_j^k) \quad \forall i = 1, 2, \dots, n \quad (4.11)$$

where  $\Omega_{kj}$  was defined in Equation 4.9 and the operator  $\otimes|$  denotes element-by-element AND-ing of the resulting  $n - 1$   $\Omega_{ij} \underline{\Omega}_j^k$  vectors.<sup>5</sup> Note  $\Omega_{ij}$  is constant and may be precomputed. Equation 4.11 basically implements the LDR iteratively. When  $\underline{\Omega}_j^{k+1} = \underline{\Omega}_j^k \forall j$ , the limit set is obtained.

The iterative formulation of Equation 4.11 is also very significant in that it suggests a nonbinary extension incorporating the notion of *label support*. This is treated in Section 4.10.

### Relaxation Implementation Using MATLAB/Octave

The formulation of Equations 4.9 and 4.11 in MATLAB/Octave is straightforward and facilitated by the availability of multidimensional arrays.

<sup>5</sup> Here we adopt the MATLAB/Octave interpretation of a binary element having value 0 or some nonzero value.

```

/* label2b-time.pro */
/* just one of two unary constraints (highest) */
/* addition of write and fail to print all solutions */
/* timing version */

highest(sky).
adj-to(car,road).
adj-to(road,grass).
adj-to(road,trees).
adj-to(sky,trees).
adj-to(grass,trees).

adjacent-to(R1,R2) :- adj-to(R1,R2).
adjacent-to(R1,R2) :- adj-to(R2,R1).

goal :- highest(R1),
        adjacent-to(R1,R2),
        adjacent-to(R2,R6),
        adjacent-to(R2,R3),
        adjacent-to(R3,R4),
        adjacent-to(R2,R4),
        adjacent-to(R4,R6),
        adjacent-to(R4,R5),
        write('Solution is'),nl,nl,
        write('region 1 is bound to '),write(R1),nl,
        write('region 2 is bound to '),write(R2),nl,
        write('region 3 is bound to '),write(R3),nl,
        write('region 4 is bound to '),write(R4),nl,
        write('region 5 is bound to '),write(R5),nl,
        write('region 6 is bound to '),write(R6),nl,nl,
        statistics(inferences,Number),write(Number),nl,
        fail.

```

Version without limit set

```

/* label-limit-set-time.pro */
/* uses limit set to prefilter solutions and
   reduce thrashing */
/* Gamma_1 determined from Gamma_1=[sky] (highest) */
/* addition of write and fail to print all solutions */
/* timing version */

adj-to(car,road).
adj-to(road,grass).
adj-to(road,trees).
adj-to(sky,trees).
adj-to(grass,trees).

adjacent-to(R1,R2) :- adj-to(R1,R2).
adjacent-to(R1,R2) :- adj-to(R2,R1).

```

```

goal :- gamma1(Gamma1), member(R1, Gamma1),
       gamma2(Gamma2), member(R2, Gamma2),
       gamma3(Gamma3), member(R3, Gamma3),
       gamma4(Gamma4), member(R4, Gamma4),
       gamma5(Gamma5), member(R5, Gamma5),
       gamma6(Gamma6), member(R6, Gamma6),
       adjacent-to(R1,R2),
       adjacent-to(R2,R6),
       adjacent-to(R2,R3),
       adjacent-to(R3,R4),
       adjacent-to(R2,R4),
       adjacent-to(R4,R6),
       adjacent-to(R4,R5),
       write('Solution is'),nl,nl,
       write('region 1 is bound to '),write(R1),nl,
       write('region 2 is bound to '),write(R2),nl,
       write('region 3 is bound to '),write(R3),nl,
       write('region 4 is bound to '),write(R4),nl,
       write('region 5 is bound to '),write(R5),nl,
       write('region 6 is bound to '),write(R6),nl,nl,
       statistics(inferences,Number),write(Number),nl,
       fail.

```

```

/* the (previously determined) limit set */

gamma1([sky]).
gamma2([trees]).
gamma3([road,grass]).
gamma4([road,grass]).
gamma5([trees,road,car,grass]).
gamma6([road,grass]).
```

Version incorporating limit set

**Figure 4.15** Original and Limit-Set-Modified Prolog Solution Formulations with Run-Time Statistic Monitoring

```

?- ['label2b-time.pro'].
% label2b-time.pro compiled 0.00 sec, 2,788 bytes

Yes
?- goal.
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to road
region 4 is bound to grass
region 5 is bound to trees
region 6 is bound to road

219642
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to road
region 4 is bound to grass
region 5 is bound to road
region 6 is bound to road

219668
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to grass
region 4 is bound to road
region 5 is bound to grass
region 6 is bound to grass

219791
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to grass
region 4 is bound to road
region 5 is bound to trees
region 6 is bound to grass

219816
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to grass
region 4 is bound to road
region 5 is bound to car
region 6 is bound to grass

219842
No
?- ['label-limit-set-time.pro'].
% label-limit-set-time.pro compiled 0.00 sec, 4,316 bytes

Yes
?- goal.
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to road
region 4 is bound to grass
region 5 is bound to trees
region 6 is bound to road

7261
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to road
region 4 is bound to grass
region 5 is bound to road
region 6 is bound to road

7331
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to grass
region 4 is bound to road
region 5 is bound to trees
region 6 is bound to grass

7512
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to grass
region 4 is bound to road
region 5 is bound to car
region 6 is bound to grass

7626
Solution is

region 1 is bound to sky
region 2 is bound to trees
region 3 is bound to grass
region 4 is bound to road
region 5 is bound to grass
region 6 is bound to grass

7694
No
?-

```

Figure 4.16 Run-Time Statistics for Original and Limit-Set-Modified Prolog Solutions

**Octave Code and Results.** A straightforward implementation of the iterative solution approach is shown in Figure 4.17. Sample results for two different initial unary constraint cases are shown in Figure 4.18. Case 1 only incorporates a unary constraint on  $u_1$  (sky), and indicates multiple solutions. Case 2 incorporates both unary constraints on  $u_1$  and  $u_4$  (sky and road, respectively), and leads to a unique labeling.

Observe, as the MATLAB/Octave source indicates, that the label order in the omega vectors is “car,” “grass,” “road,” “trees,” “sky.” This facilitates comparison with the handworked solutions.

```

%% label.m; iterative labelling formulation
%% image example
%% label order in omega vectors:
%%      car grass road trees sky
%% initial label sets (D_i) and unary constraints
%% enforce (only) unary 'sky' label on u1
%% requires
omega(:,1)=[0 0 0 0 1]';

%% u2-u6 can have any label

omega(:,2)=[1 1 1 1 1]';
omega(:,3)=[1 1 1 1 1]';
omega(:,4)=[1 1 1 1 1]';
omega(:,5)=[1 1 1 1 1]';
omega(:,6)=[1 1 1 1 1]';

%% display omega with unary constraints only
printf("\nInitial omega (before iteration) is: \n");
omega

%% form Pij

for i=1:6
  for j=1:6
    if(i!=j)
      P(:,:,i,j)=omega(:,i)*omega(:,j)';
    endif
  endfor
endfor

E=ones(5);

%% use A to represent adjacency constraint
%% rows/cols of A indexed from R1 (u1) to R6 (u6)

A=[0 1 0 0 0 0; 1 0 1 1 0 1
  0 1 0 1 0 0; 0 1 1 0 1 1;
  0 0 0 1 0 0; 0 1 0 1 0 0]

%% C from allowed pairwise labelling (if adjacent) graph
%% label order in omega vectors:
%%      car grass road trees sky

C=[0 0 1 0 0;
  0 0 1 1 0;
  1 1 0 1 0;
  0 1 1 0 1];

```

```

0 0 0 1 0]
%% form omegaij

for i=1:6
  for j=1:6
    if(i!=j)
      omegaij(:,:,:,i,j)=P(:,:,:,:,i,j) & ((!A(i,j)*E)|C);
    endif
  endfor
endfor

%% iteratively update ui using omegaij and ui

count=0;

while(1)
  count=count+1;
  %% store old omega
  omega_old=omega;

  %% get updated omega ui from uj; j!=i
  %% this updates uses the newest ui for each update,
  %% i.e., asynchronously

  tmp=ones(5,1);

  for i=1:6
    for j=1:6
      if(i!=j)
        tmp = (omegaij(:,:,:,i,j)*omega(:,j)) & tmp;
      endif
    endfor
    omega(:,i) = tmp;
    tmp=ones(5,1);
  endfor

  %% check for convergence

  if(omega==omega_old)
    printf("\nfinal omega is: \n");
    omega
    break
  endif

  %% display new omega at iteration count
  printf("\nomega at iteration %d is: \n",count);
  omega
endwhile

```

Figure 4.17 Iterative Image Labeling Formulation Using MATLAB/Octave



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



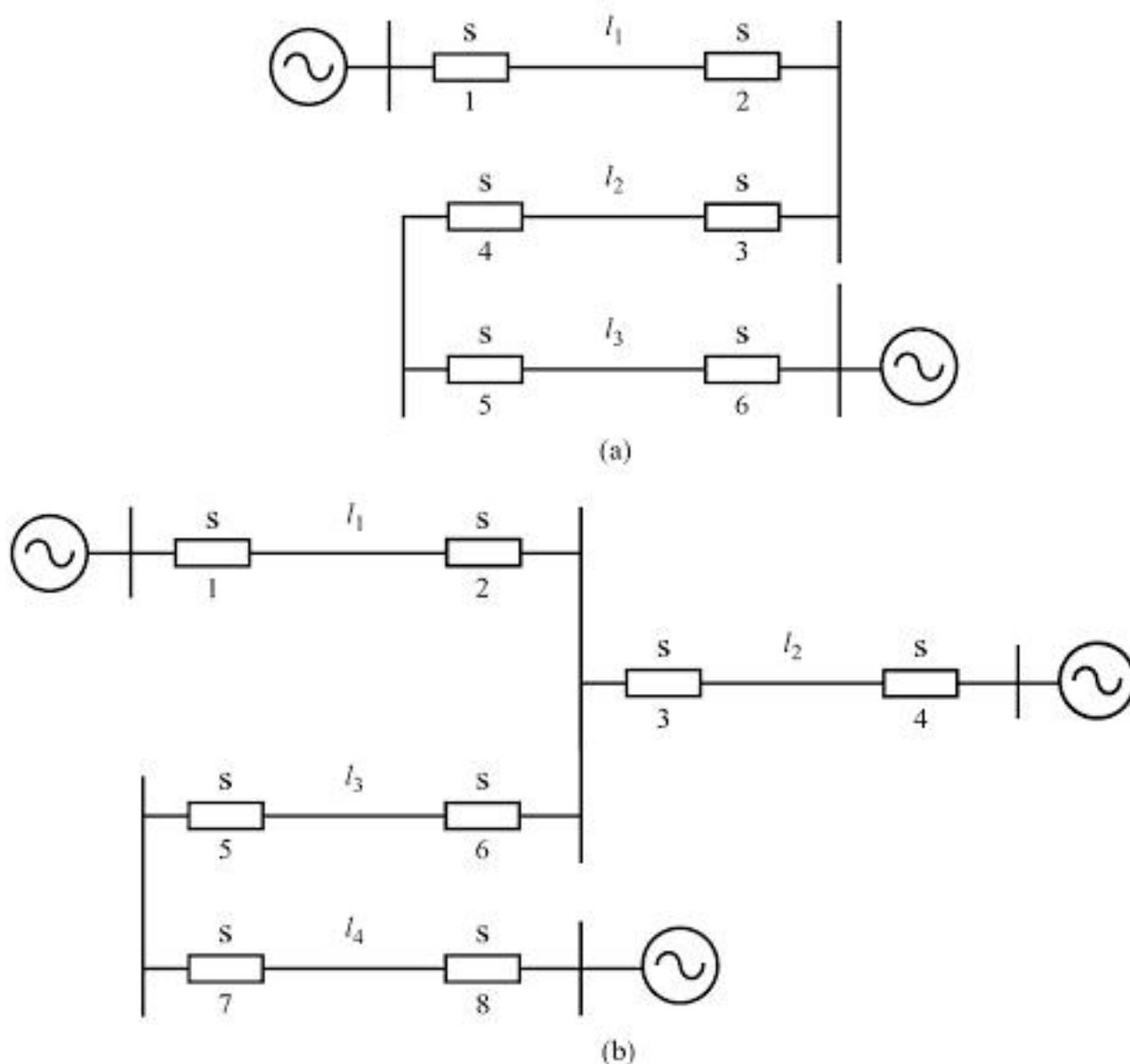
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



**Figure 4.25** Power System Topology Used for “generation” Examples

### Backup Breaker Operation Constraints and Prolog Description

The role of a backup breaker is described in the Prolog database. For example, predicate

```
back_up(B1,B2)
```

is used to represent the fact that Breaker B2 is one of B1’s backup breakers. Note that B1 may have any number of backup breakers, which are a function of the power network topology. Backup operation is specified via the constraint:

```
back_up(B1,B2) :- not(generation(B1)), /* no backup for generators */
connection(B1,B3),
other_breaker(B3,B2),
has_gen(B2).
```

In assessing protection system operation, the rule `backup_did_not_work(B1,B2)` signifies that back-up breaker, B2, of Breaker B1 did not operate as a back-up breaker. This constraint, formulated as a Prolog rule, is:

```
backup_did_not_work(B1,B2) :- back_up(B1,B2),
                                not(operate(B2)).
```

### Determination of Fault Locations Through Electrical Isolation

In order to determine a possible fault location, we assume that electrically isolated lines (i.e., those with generation cutoff) are candidates for fault locations. This is done via rule `no_source_coming(B1)` that is true when the generation source is cut off from the side of the line containing Breaker B1. This constraint leads to the three Prolog-rule set:

```
no_source_coming(B1) :- not(has_gen(B1)).
no_source_coming(B1) :- has_gen(B1),
                        operate(B1).
no_source_coming(B1) :- back_up(B1,_),
                        not(backup_did_not_work(B1,_)).
```

which is used for this specification.

The heart of the matter, in this CSP, is that *electrically isolated lines are candidate fault locations*. Prolog predicate `elect_isolated` is used to represent this constraint and thus determine possible fault locations and corresponding interpretations. The Prolog rule constraining a line containing a fault location to have no generation is:

```
elect_isolated(L,B1,B2) :-
    no_source_coming(B1),
    no_source_coming(B2), !.
```

### Rules for Interpretation and Display

Rule `printout` gives the interpretation of correct and incorrect (primary) breaker operation:

```
printout(B) :- has_gen(B),
              operate(B),
              write('Breaker '),
              write(B),
              write(' operated correctly.'), nl, !.
```

```
printout(B) :- has_gen(B),
              not(operate(B)),
              write('Breaker '),
              write(B),
              write(' malfunctioned.'), nl,
              not(printbackup(B)), !.
```

For output, the rule “`printbackup(B)`” prints all correct operations corresponding to backup breakers. Note it always evaluates to FALSE due to the final fail predicate in the body. This is a way

to force the Prolog unification mechanism to exhaustively find, and enumerate, all possible successful unifications. The rule is:

```
printbackup(B) :-  
    back_up(B,B1),  
    operate(B1),  
    write('Breaker '),  
    write(B1),  
    write(' operated correctly as a back-up  
breaker.'),  
    nl,fail.
```

Finally, we incorporate our overall goal in the database:

```
run :- protected_by(L,B1,B2),  
      elect_isolated(L,B1,B2),  
      write('Possible Fault Location is on line '),  
      write(L),nl,  
      printout(B1),  
      printout(B2),  
      nl,nl,  
      fail.
```

Notice again the use of fail, which, together with the predicate `protected_by`, as well as the topological description, forces an assessment of the status of each line.

#### 4.12.8 Listing of Overall Prolog Program

Note this listing contains the topology corresponding to Figure 4.24.

```
/* file: power1.pro
```

```
This program finds the fault locations in a power system  
network and gives the interpretation of the breaker operation.
```

Key assumptions are:

- 1) only one possible breaker malfunction,
- 2) the malfunctioned breaker has to have backup breakers.
- 3) only line fault is possible,
- 4) the relay has direction characteristic.

The database consists of:

- 1) topology of the network,
- 2) the current breaker status,
- 3) rules,
- 4) goal. \*/

```

/* 1) topology data for system */

protected_by(line1,1,2).
protected_by(line2,3,4).
protected_by(line3,5,6).
protected_by(line4,7,8).

connect(2,3).
connect(2,6).
connect(2,7).
connect(3,6).
connect(3,7).
connect(6,7).

generation(1).
generation(4).
generation(5).

/* 2) breaker status */

operate(1).
operate(4).
operate(5).

/* 3) rules for breaker operation and interpretation */

connection(B1,B2) :- connect(B1,B2).
connection(B1,B2) :- connect(B2,B1).

other_breaker(B1,B2) :- protected_by(_,B1,B2).
other_breaker(B1,B2) :- protected_by(_,B2,B1).

has_gen(B) :- generation(B), !.
has_gen(B) :- connection(B,B1),
            other_breaker(B1,B2),
            has_gen(B2), !.

back_up(B1,B2) :- not(generation(B1)), /* no backup for generators */
                connection(B1,B3),

```



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



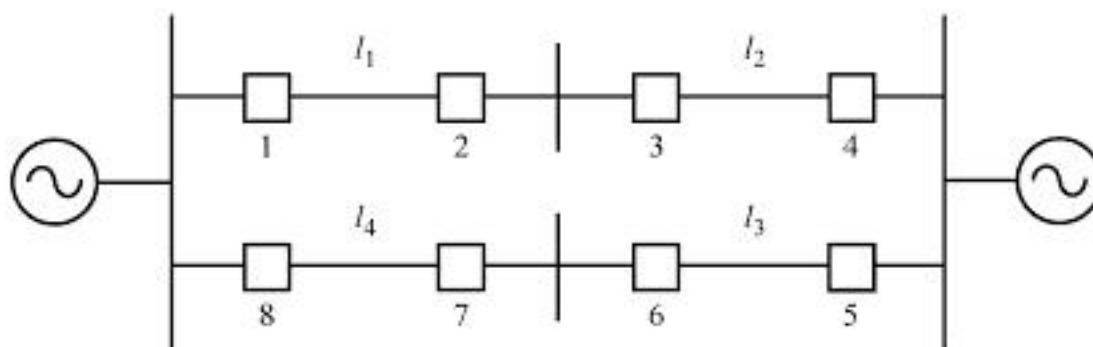
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



**Figure 4.29** Modified System Topology for Power System Protection Interpretation CSP

28. Repeat Problem 27 for each of the topologies in Figure 4.30(a) with the following cases of breaker status:

Breaker	Case 1	Case 2	Case 3
1	S	S	O
2	S	S	S
3	S	O	O
4	S	S	S
5	O	S	S
6	O	O	S

29. Repeat Problem 27 for each of the topologies in Figure 4.30(b) with the following cases of breaker status:

Breaker	Case 1	Case 2	Case 3
1	O	O	S
2	S	S	S
3	S	S	S
4	O	S	S
5	S	S	O
6	S	S	S
7	S	S	S
8	S	O	O

30. In the power protection CSP of Section 4.12, a somewhat “negative” formulation of several constraints (especially concerning backup operation) was shown. Examples are:

```
backup_did_not_work(B1,B2) :- back_up(B1,B2),
    not(operate(B2)).
```

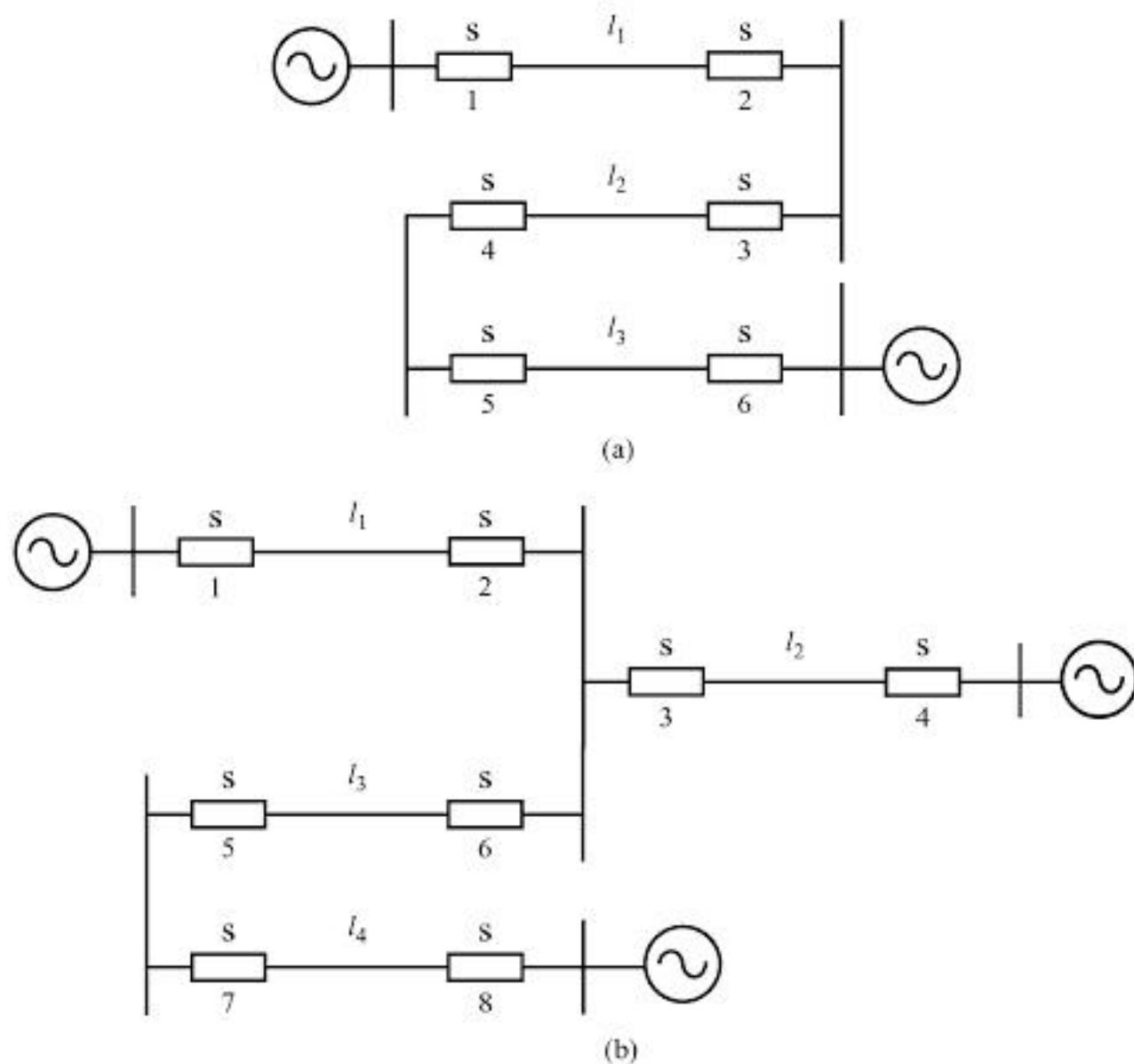


Figure 4.30 Additional Power System Topologies for CSP

and

```
no_source_coming(B1) :- back_up(B1,_),
    not(backup_did_not_work(B1,_)).
```

Investigate the possibility and consequences of converting this representation to a more positive formulation, i.e., one without the use of the `not` predicate.

31. Discuss the feasibility of implementing CSP solutions in parallel. (This is a very open-ended problem.)

# CSPs, Part 2: Structural Approaches Leading to Natural Language Understanding and Related Topics

## 5.1 Introduction

In Chapter 4, we explored an important class of constraint satisfaction problems common to many IS applications. In this chapter we extend our look at constraint satisfaction-based IS formulations and consider CSPs that involve structural, or *structure-based* constraints. This includes natural language (NL) understanding, an important IS application area.

## 5.2 What Types of Problems Have Structural Constraints?

Our examination of modeling in Chapter 1 indicated that many IS applications exhibit a strong dependence on a clearly defined structural model. Most noteworthy, and emphasized in this chapter, is natural language. However, many other problem domains have structural concerns. As we show, some of the language structure-based modeling and solution approaches (e.g., grammars) are also applicable in these domains. Examples include:

1. Music, as shown in the following paragraph;
2. Line drawings, as shown in Figure 5.1;
3. Signals, such as the cardiac EKG signal shown in Figure 5.2; and
4. Mechanical assemblies, typically shown in “exploded views” or drawings.

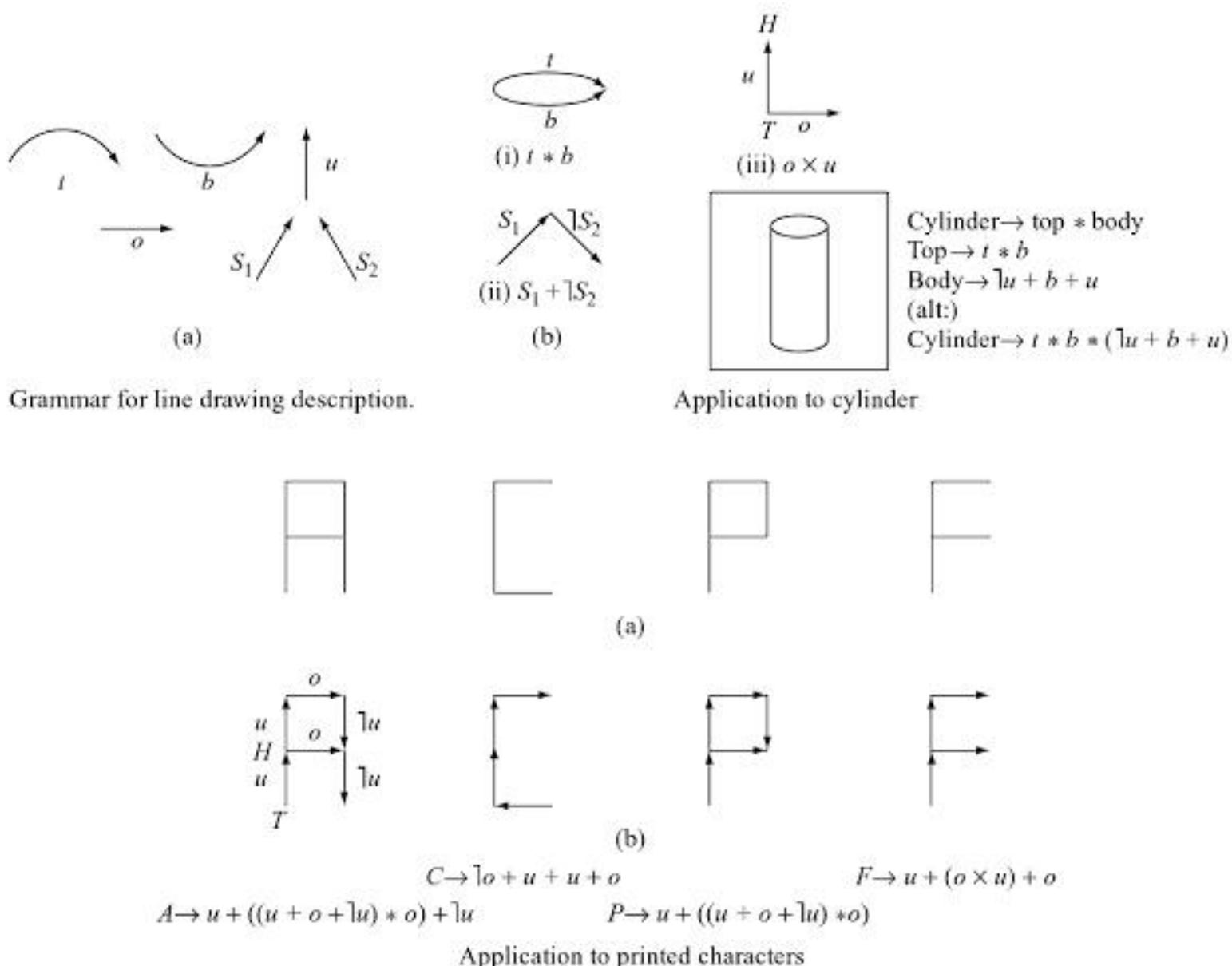


Figure 5.1 Structural Interpretation of Line Drawings

**Structure, Computer Music, and CSPs.** Open Music [Ope] is a visual programming and development environment for computer-aided music composition. As shown in the example of Figure 5.3, Open Music facilitates constraint programming and allows music composition to be treated as a CSP. Constraint satisfaction formulations can be defined and integrated into Open Music using a graphical interface. They may be solved using a number of different constraint solvers, including the Oz<sup>1</sup> constraint language [HLZ96]. Oz provides a programming interface that allows the interactive exploration of search trees and implementation of parallel search.

<sup>1</sup><http://www.mozart-oz.org/>



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



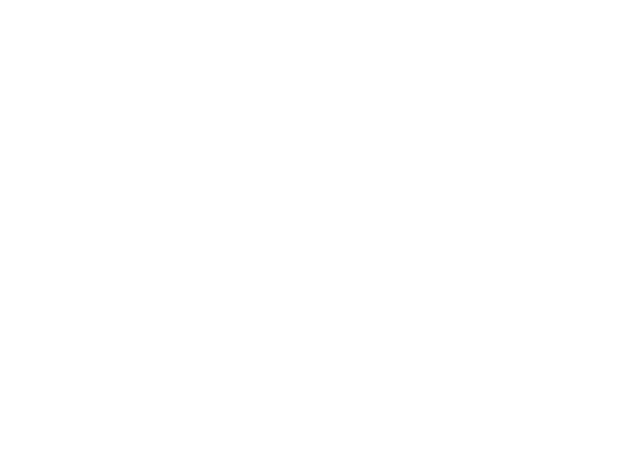
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

```

/* here's a sentence without semantic constraints --
sentence([Noun,Verb,Object]) --> np(Noun), vp([Verb, Object]).
*/

/* use the above semantic constraints */

sentence([Subject,Verb,Object]) --> np(Subject), vp([Verb, Object]),
{agree(Subject,Verb,Object)}.

/* more than just simple enumeration
but short of a structured representation */

agree(Something,Action,What) :- Something \= What, % no reflexive
Something=program,
(Action=runs; Action=crashes),
checkObject(Action,What).

agree(Something,Action,What) :- Something \= What,
Something=computer,
Action=runs,
checkObject(Action,What).

checkObject(Action,What) :- (Action=runs; Action=crashes),
What=computer.

checkObject(Action,What) :- Action=runs,
(What=program; What=plant).

/* rest is unaffected except no longer lists */

np(Noun) --> adj, noun(Noun).
vp([Verb, Object]) --> verb(Verb), np(Object).

/* slightly reduced 'dictionary' of terminals */

adj --> [the].

noun(program) --> [program].
noun(computer) --> [computer].
noun(plant) --> [plant].

verb(crashes) --> [crashes].
verb(runs) --> [runs].

```

If the preceding constraints are implemented, using the Prolog parser/generator as shown, the resulting language is:

```

?- ['generate-lgn2-agree2.pro'].
% generate-lgn2-agree2.pro compiled 0.00 sec, 128 bytes

Yes
?- sentence(Structure,What,[]).

```

```

Structure = [program, crashes, computer]
What = [the, program, crashes, the, computer] ;

Structure = [program, runs, computer]
What = [the, program, runs, the, computer] ;

Structure = [program, runs, plant]
What = [the, program, runs, the, plant] ;

Structure = [computer, runs, program]
What = [the, computer, runs, the, program] ;

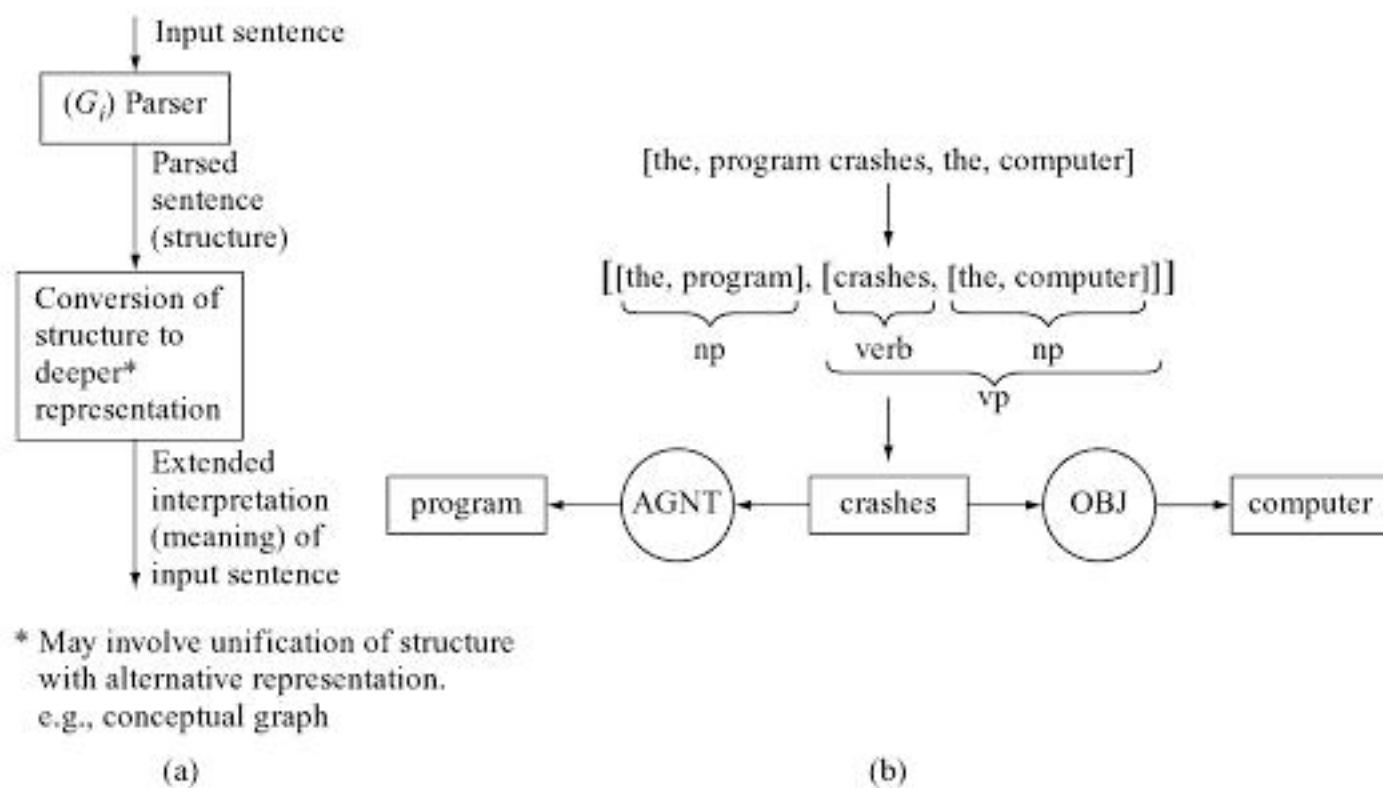
Structure = [computer, runs, plant]
What = [the, computer, runs, the, plant] ;

```

No

## 5.7 From a Simple Parse to Structure to Meaning

Once the sentence has been successfully parsed (and the higher-level structure is evident from the parse), the deeper meaning of the sentence may be determined. This is shown in Figure 5.7.



**Figure 5.7** Use of Sentence Structural Information in Understanding. (a) Steps (b) Hypothetical Conversion from Parser-Provided Structure to Representation of Meaning

## 5.8 Feature Structure-Based Representation and Manipulation

This chapter (as well as Chapter 4) emphasize IS situations wherein we seek solutions where “things fit.” In more quantitative terms, a CSP is satisfied. Specific examples included CSP (labeling) problems as well as problems with structural constraints imposed via phrase structure grammars.

### 5.8.1 Motivations

In this section, we consider the generalizations and extensions of previous strategies for structural CSPs. This leads to strategies based upon feature structures (FS).

#### The Limitations of Context-Free Grammars and Term Unification

CFGs are not quite powerful enough for use with natural languages. We are, however, interested in retaining some notion of unification as a general processing strategy. To do this, we seek to define and implement a more general type of representation and unification. One strategy is to look at *Unification Grammars* and *Feature Structures*. This yields a framework for computational semantics that is suitable for parsing and generation.

#### The Limitations of Prolog and First-Order Logic

The built-in unification mechanism in Prolog is quite useful. Recall, however: *Prolog implements first-order logic*. As simplistic as this statement may appear, it alludes to our interest in higher-order logics and alternative representations. The principal constraint on a first-order logic is that *variables may not be used as placeholders for predicate names*. This is more than an issue of representational convenience; the concept of unification when the predicate names are not constant raises a number of important theoretical issues. As shown in Section 5.8.2, there are potential “workarounds” to allow this type of Prolog representation, however the basic issue of how unification should proceed in this case is complex.

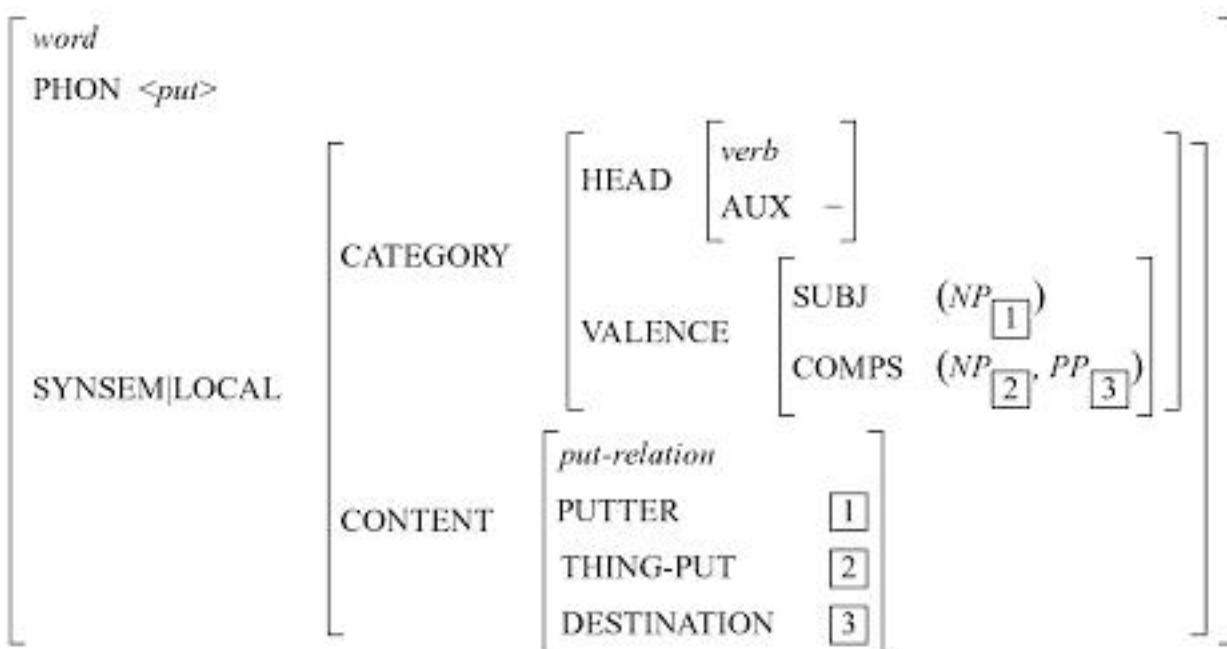
#### An Extension: The Head-Driven Phrase Structure Grammar

The Head-Driven Phrase Structure Grammar (HPSG) is an approach to grammatical theory that seeks to model human languages more generally as systems of constraints [CFPS06, LM]. Typed FS (defined in the following section) have a central role in this grammatical model. HPSG research is conducted by a consortium of research institutions. Good starting points are <http://hpsg.stanford.edu/> and <http://www.ling.ohio-state.edu/research/hpsg>. An example of the representation of the verb “put” in the HPSG is shown in Figure 5.8.

### 5.8.2 Feature Structures (FS) and Basic Properties

#### Unification Grammars and Feature Structures

Unification grammars encode knowledge as a declarative specification of constraints. Within the unification-based grammar framework, a complex data structure referred to as a feature structure



**Figure 5.8** HPSG Representation of the Verb “put” (adapted from [ref], tutorial available at <http://purl.org/dm/papers/all2-hpsg.pdf>)

<i>Feature<sub>1</sub></i>	<i>Value<sub>1</sub></i>
<i>Feature<sub>2</sub></i>	<i>Value<sub>2</sub></i>
...	
<i>Feature<sub>n</sub></i>	<i>Value<sub>n</sub></i>

**Figure 5.9** Very Simple (“Hello World”) FS to Illustrate FS Concept

[Kni89, Shi86], or functional description [Kay85], encodes information from various sources as feature/value pairs. Perhaps the simplest FS is shown in Figure 5.9. By noting that values may themselves be FS, more complex (and useful) FS are possible.

**FS: Formal Definition.** A feature structure may be viewed as a partial function from features to their values [Shi86]. For instance, a function mapping the feature object onto the value *house* and mapping viewpoint onto *top* would be represented in a feature structure notation as shown in Figure 5.10. More complex feature structures contain partial functions from features to values that are themselves feature structures, as shown in Figure 5.11.

**FS: Related Definitions and Properties.** Feature structures are related to frames (Chapter 8) and are similar in appearance to “record”-like structures in some programming languages. They are useful as a data structure in certain CSPs, especially when operations such as generalization, specialization, and unification are desirable. FS may quantify concepts, as well as instances of these concepts. FS are also potentially useful in information fusion.

### Relationship to First-Order Logic

Feature structures resemble first-order logic terms but have several important distinctions [Kni89]. The first difference is that feature substructures are labeled symbolically. First-order terms require strict ordering of information; feature structures lift this restriction by labeling the information and,



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

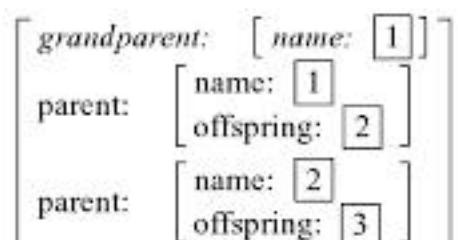


Figure 5.19 “Grandparent” Rule as FS

### The Infinite Lattice, $L_I$

The *infinite lattice*, denoted  $L_I$ , is the representation that would exist if all possible models and elements that could be modeled within a physical world were modeled. However, there will always exist more elements than can be physically represented within a computer vision system (a model is always incomplete) and, therefore, this lattice is impossible to realize. The infinite lattice is the conceptual whole from which the subset of elements to be modeled in a computer vision system will be taken.

### The Enumerated Lattice, $L_E$

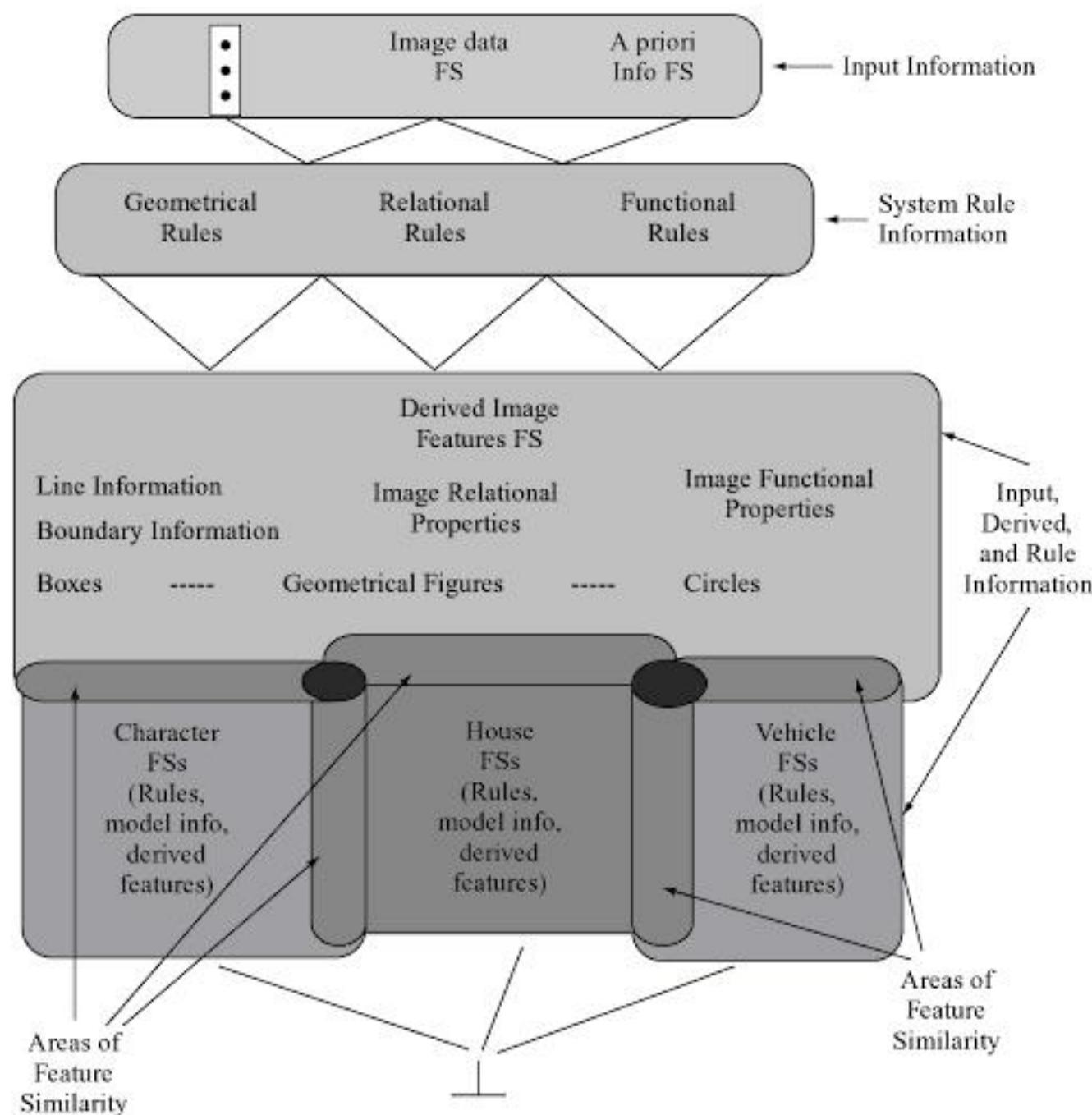
The *enumerated lattice*, denoted  $L_E$ , is the lattice that, in theory, could be produced given the entire set of FS object models, FS rules, and FS inputs modeled. This lattice consists of all FS unification paths that are possible, given the set of objects and object components modeled as FS and the set of rules expressed as FSs. Within sections of the enumerated lattice, locality of information is readily apparent. As shown in Figure 5.20, portions of the enumerated lattice can be identified as containing information about things such as houses, cars, or character information. These areas may overlap where there are areas of common features, such as boxes or circles or the common structure of line information. This locality of information within the enumerated lattice is beneficial in partitioning the rules expressed as FSs into sets for the purposes of unification and search.

### The Search Lattice, $L_S$

The *search lattice*, denoted  $L_S$ , is the lattice that is actually produced in the search through the enumerated lattice during the determination of image content. This lattice contains only the FSs and FS unification paths that are produced in the current search through the enumerated lattice to reach a goal. The search lattice is the subset of the enumerated lattice explored during the search to the goal state.

## 5.8.7 Lattice Processing Directions

One of the most significant and open problems in using FSs is choice of what rules to use and what FSs to unify. This is our old friend conflict resolution, and FSs, in themselves, do little to facilitate this.



**Figure 5.20** Example of Possible Arrangement of Features within a Lattice (image processing/computer vision application)

### The Search Problem Within the Lattice of Feature Structures

Search is one of the central issues in problem-solving systems [TB92]. It occurs whenever the system is faced with a choice from a number of alternatives, where each choice leads to the need to make further choices, and so on until the problem is solved.

A general search problem is characterized by the following [Par91, Sch90]: a set of system states that represent the problem state space, a set of initial system states contained in the set of system states, a set of transformations (operators, rules) that can transform one state into another, and a set of goal states contained in the set of system states. Thus, problem solving is represented as finding and applying a sequence of operators that will transform an initial system state, through a sequence of intermediate states, into a system goal state.

Any problem represented by the lattice of FSs still embodies a general search problem. The set of system states that represent the problem state space are contained in  $L_E$ . The initial system state is the input data and a priori information input into the system as a feature structure. The set of transformations (operators, rules) that can transform one state into another are the rules expressed as FSs within the enumerated lattice. Finally, the set of goal states contained in the set of system states could be the world models of the objects or scenes possibly contained in the input image. The search problem consists of finding and applying a sequence of FS rules that will transform initial data and a priori information (initial system state) into a system goal state containing the explanation of the input data as an expanded FS.

A possible solution path through the lattice, if it exists, may be visualized for small problems; however, with even modest problems, there are many possible search lattices. The decision of which FSs to unify in the production of a search lattice must be chosen from many possible applicable feature structures in the system. The critical problem-solving step is to determine which grammar rules expressed as FSs are appropriate at the given time and then decide which FSs may be unified with the chosen grammar rule.

### FS-Based Generalization

The option of generalization is also available in the search through the lattice. Generalization is often referred to as the dual of unification [Kni89]. Within the lattice framework, unification corresponds to finding the GLB of two FSs contained in the lattice. Alternately, generalization corresponds to finding the LUB of two FSs contained in the lattice. The top of the lattice ( $T$ ), to which all pairs of terms can generalize, is also called the *universal term* or *universal variable*. Every feature structure contained in the lattice is an instance of this term [Kni89].

Generalization is used to point to common aspects within data and also point to other promising solution paths to explore. Sample generalization operators are shown in Chapter 16, Section 16.3.11. They include the dropping condition operator and the constants to variables operator. The dropping condition operator relaxes conditions required by a grammar rule FS. The constants to variables operator relaxes required values contained in the FS models to variables. In this method, general aspects of a world model or FS grammar rule are retained, but some specifics of the model or rule may be relaxed to allow the unification search path through the lattice to proceed in other directions so that a viable solution may be determined.

## 5.9 Exercises

1. The list-based output from the Prolog sentence generator and parser is acceptable, but perhaps a more familiar output could be generated. Consider a Prolog predicate to print the resulting sentence (still without capitalization and a period). Suppose we have:

```
alist([my,dog,has,fleas]).  
goal :- alist(X), printAlist(X).
```

The desired behavior is as follows:

```
?- goal.  
my dog has fleas
```

Yes

2. Modify the Prolog-based language generator and parser for the (*G*1) grammar such that questions are generated and parsed. First, develop the revised grammar, then the Prolog implementation. An example use is shown below:

```
?- question([can, the, program, crash, a, computer], []).
```

Yes

3. Modify the grammar of Section 5.6.4 to enable formation and recognition of compound sentences using the connectives “and,” “or,” and “but.” For example, the following should be generated and parsed:

```
[a, program, crashes, a, program, and, a, program, crashes, a, program]  
[a, program, crashes, a, program, and, a, program, crashes, a, system]  
[a, program, crashes, a, program, and, a, program, crashes, a, computer]
```

.

```
[a, computer, runs, the, program, and, a, program, crashes, a, system]  
[a, computer, runs, the, program, and, a, program, crashes, a, computer]  
[a, computer, runs, the, program, and, a, program, crashes, the, program]
```

.

```
[the, program, runs, a, computer, but, the, program, runs, the, system]  
[the, program, runs, a, computer, but, the, program, runs, the, computer]  
[the, program, runs, a, computer, but, the, system, crashes, a, program]
```

4. In the preceding problem, can you design a grammar and a corresponding Prolog implementation such that longer (multiple connectives are used) compound sentences are generated?

5. Modify the *G*1 generator/parser so paragraphs are generated and recognized. Recall a paragraph should be longer than one sentence.

6. A student attempted to solve the *G*1 meaningful semantics (context) problem with the following solution:

```
sentence([Subject,Verb,Object]) --> np(Subject), vp([Verb, Object]),  
                                {agree(Subject, Verb, Object)}.

agree(program, runs, What) :- What \= program.
agree(program, crashes, What) :- What \= program.

agree(Something, run, program) :- Something \= program.
```



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

- (b)  $\left[ \begin{array}{ll} \text{Number} & \text{singular} \end{array} \right] \sqcup \left[ \begin{array}{ll} \text{Number} & \text{singular} \end{array} \right]$
- (c)  $\left[ \begin{array}{ll} \text{Number} & \text{singular} \end{array} \right] \sqcup \left[ \begin{array}{l} \text{Number} \\ [] \end{array} \right]$
- (d)  $\left[ \begin{array}{ll} \text{Number} & \text{singular} \end{array} \right] \sqcup \left[ \begin{array}{ll} \text{Number} & \text{plural} \end{array} \right]$
- (e)  $\left[ \begin{array}{ll} \text{Number} & \text{singular} \end{array} \right] \sqcup \left[ \begin{array}{ll} \text{Person} & \text{first} \end{array} \right]$
- (f)  $\left[ \begin{array}{ll} \text{Number} & \text{singular} \end{array} \right] \sqcup \left[ \begin{array}{l} \left[ \begin{array}{ll} \text{Number} & \text{singular} \end{array} \right] \\ \left[ \begin{array}{ll} \text{Person} & \text{first} \end{array} \right] \end{array} \right]$
- (g)  $\left[ \begin{array}{l} \text{pizza} \\ \text{CRUST thick} \\ \text{TOPPINGS} \left[ \begin{array}{l} \left[ \begin{array}{l} \text{OLIVES YES} \end{array} \right] \\ \left[ \begin{array}{l} \text{ONIONS NO} \end{array} \right] \end{array} \right] \end{array} \right] \sqcup \left[ \begin{array}{l} \text{pizza} \\ \text{CRUST thick} \\ \text{TOPPINGS} \left[ \begin{array}{l} \left[ \begin{array}{l} \text{OLIVES YES} \end{array} \right] \\ \left[ \begin{array}{l} \text{HAM NO} \end{array} \right] \end{array} \right] \end{array} \right]$
- (h)  $\left[ \begin{array}{l} \text{rig:} \left[ \begin{array}{l} \text{manufacturer: bic} \\ \text{mast: excel} \\ \text{boom: bic} \\ \text{sail: neil pryde} \\ \text{harness: none} \end{array} \right] \end{array} \right] \sqcup \left[ \begin{array}{l} \text{rig:} \left[ \begin{array}{l} \text{manufacturer: bic} \\ \text{length: 328cm} \\ \text{volume: 190L} \\ \text{footstraps: none} \end{array} \right] \end{array} \right]$

15. Verify the FS unification shown in Figure 5.14 using:

- (a) The definition of subsumption; and
- (b) The concept of a LUB.

16. The purpose of this problem is to compare and contrast the feature structure paradigm with two other IS technologies, namely:

- (a) Ontologies (Chapter 2); and
- (b) Frames (Chapter 8).

Specifically, we are interested in the similarities and differences among the three representational schemes (and their corresponding manipulation strategies, where applicable). Develop a table for comparison. What can you do with one that cannot be done with others? For example,<sup>7</sup> you might address the following issues:

- (a) Consider the concept of *inheritance* as it might apply to each.
- (b) How is each manipulated?
- (c) Can each of these representational schemes be easily modified or updated?

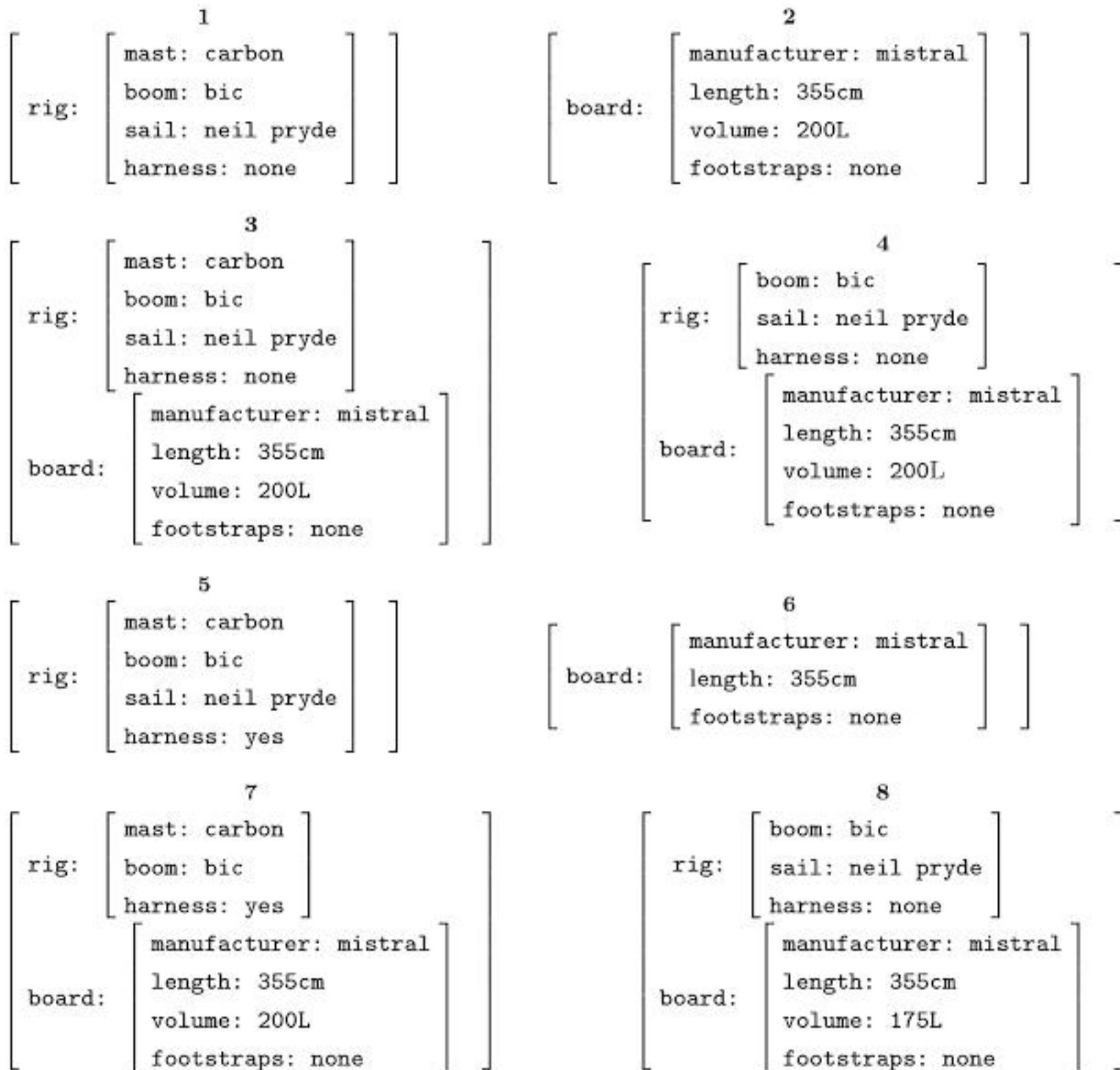
17. Suppose FS *a* and FS *b* are unifiable and their unification yields FS *c*. Must *c* be unique? (Hint: This may be recast as a question regarding the uniqueness of the LUB.)

<sup>7</sup>This is not all-inclusive.

18. This chapter considered the pairwise attempted unification of FS. A reasonable question is whether this is necessary, i.e., can we consider the unification of more than two FS? Specifically, suppose we are interested in determining  $a \sqcup b \sqcup c$ . Consider several viewpoints:

- (a) Successive pairwise unification (e.g.,  $a \sqcup b$  and  $b \sqcup c$ , followed by unification of the results).
- (b) Observation that a LUB is defined for a set of elements with cardinality greater than 2.
- (c) Use of the subsumption definition.

19. Each FS below is “indexed” by a bold numeric label above the FS. This has nothing to do with co-reference, but just makes referring to the FS easier. *Use this label in place of rewriting each FS in this problem.* Using the relation “subsumes,” arrange the following FS in a lattice.



9		10
rig: [ mast: carbon boom: bic harness: yes ]		rig: [ boom: bic sail: neil pryde harness: none ]
board: [ length: 355cm volume: 200L footstraps: none ]		board: [ manufacturer: mistral volume: 175L footstraps: none ]

20. Refer to Figure 5.17, which shows the portion of a “house” lattice. Consider now the FS shown in Figure 5.21. As the notes indicate, curly brackets ({} ) indicate disjunction. The disjunctive feature’s value may be any one of the structures inside the brackets.

[ object: house viewpoint: top location: { subdivision, isolated } ]
--

Figure 5.21 House Feature Structure Using Disjunction

Show, clearly and unambiguously, where the FS of Figure 5.21 would be inserted into the lattice of Figure 5.17.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

- 17.** Suppose your company produces an elegant rule-based software product wherein rules are required to be of the form:

$$(p_1 \cap p_2 \cap \dots \cap p_n) \rightarrow q$$

where the  $p_i$  and  $q$  are simple statements. The system uses forward chaining.

You discover an exciting and lucrative application, but rules in this application are not in the preceding form. Specifically, two new forms for (assumed true) rules are proposed. In both forms,  $a$ ,  $b$ ,  $c$ , and  $d$  are simple statements.

**Case 1:**

$$(a \cup b) \rightarrow (c \cup d)$$

**Case 2:**

$$(a \cup b) \rightarrow (c \cap d)$$

Determine, using logic, if the rule forms in Cases 1 or 2 may be converted into a form for which you can use your product.

- 18.** Benchmarking of rule-based systems, especially when large rule and fact bases are involved, periodically receives attention. Two well-known, and perhaps well-worn, benchmarks are “Miss Manners” and the “Waltz” benchmark. The former yields a CSP (Chapter 4) wherein Miss Manners has invited 16, 32, 64, or 128 guests with various hobbies to a dinner party. She wants to arrange the guest seating so that genders alternate and each guest will have someone on the left or right with a common hobby. The Waltz benchmark involves labeling the lines in a two-dimensional drawing with the constraint that these lines are the projection of edges from a three-dimensional object.

Using the Web as a resource, find and explore implementations of each of these benchmarks.



# The C Language Integrated Production System (CLIPS)

## 7.1 CLIPS: Conceptual Background and Motivation

The C Language Integrated Production System (CLIPS) is a computational structure and language for implementing production systems. CLIPS may be viewed as a descendant of OPS5, an early Lisp-based production system. CLIPS is written in C for portability, interoperability, and speed. It is widely available and has been ported to many different operating systems.

CLIPS is well documented; this chapter merely highlights useful introductory concepts. The reader should become familiar with the documents cited in Section 7.1.5. Acquiring the CLIPS software is addressed in Section 7.8.

### 7.1.1 CLIPS History

The origin of CLIPS dates back to 1984 at NASA's Johnson Space Center, where the intent was to provide a framework for the development of expert systems. OPS5 was deemed too difficult for widespread adoption, primarily due to the underlying Lisp implementation. After initial development, CLIPS was released outside of NASA in 1986. CLIPS is now maintained independently from NASA as public domain software.

### 7.1.2 CLIPS Structure

The CLIPS structure is a somewhat expanded version of the generic PS “triad” structure shown in Chapter 6, Figure 6.6. Figure 7.1 shows this expanded structure. Note that if the “agenda,” “user interface,” and “inference engine” were combined, the triad structure reappears.

**Programming Paradigms Supported by CLIPS.** CLIPS itself supports three programming paradigms:

1. Forward chaining, rule-based;

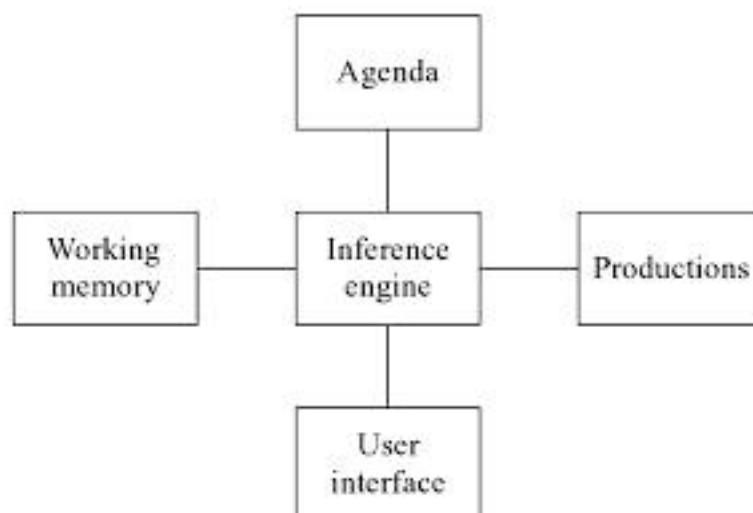


Figure 7.1 CLIPS Structure

2. Object-oriented (via COOL); and
3. Procedural.

Our emphasis is on the first paradigm, although we consider aspects of COOL; especially the integration of Protégé-developed ontologies in Chapter 8.

### 7.1.3 CLIPS Nomenclature

CLIPS uses some nomenclature that differs from other production system descriptors. The conflict set is referred to as the **agenda**. Note that CLIPS *is based on productions*, although the CLIPS documentation refers to productions as “rules.” In addition, CLIPS “facts” are stored in **wm** and may be more elaborate structures than simple facts.

### 7.1.4 Simplified CLIPS Execution Cycle

The basic execution cycle of the CLIPS IE shown in Figure 7.1 proceeds as follows<sup>1</sup>:

1. If the rule-firing limit has been reached, execution is halted.
2. If the rule-firing limit has not been reached, the top rule on the **agenda** (conflict set) is selected for execution. If there are no rules on the agenda, execution is halted. Otherwise, the right-hand side (RHS) actions of the selected rule are executed. The number of rules fired is incremented.
3. As a result of Step 2, rules may now be activated or deactivated. A new **agenda** is recomputed and the CLIPS IE returns to Step 1.

### 7.1.5 CLIPS Documentation

The accompanying CLIPS documentation consists of a number of pdf files, as shown in Table 7.1. Of these, the Basic Programming Guide should be considered mandatory reading. In addition, expert systems programming is covered in [KL96] and the use of CLIPS is shown in [GR98].

<sup>1</sup>Here we temporarily ignore the concept of focus. The **agenda** is CLIPS nomenclature for the ranked conflict set. The process of determining the **agenda** (conflict resolution) is accomplished with the Rete algorithm and discussed in Section 7.5.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

```

;; file: cardsOf.clips

;; generate 2 hands * with 5 community cards
;; via rule-based programming paradigm
;; -- using productions (instead of function(s))
;; might be simpler with a function and list data structure

;; 52 card deck template via multfield slot
;; access with nth$, use 1-52

(deftemplate deck (multislot cards)
  ; template for 'used' or previously dealt cards
  ; used cards instances in w$ initially empty

  (deftemplate dealt (slot card-value) (slot when))
    ; keep track of how many cards
  (deftemplate deal (slot cardNum))

  ; templates for 7 card hands

  (deftemplate hand1
    (slot holecard1)
    (slot holecard2)
    (slot community1)
    (slot community2)
    (slot community3)
    (slot community4)
    (slot community5)
  )

  (deftemplate hand2
    (slot holecard1)
    (slot holecard2)
    (slot community1)
    (slot community2)
    (slot community3)
    (slot community4)
    (slot community5)
  )

  ; reseed the random number generator
  ; so each run uses a newly 'reshuffled' deck
  ; must do first -- control salience

  (defrule reseed
    (declare (salience 1000))
    (initial-fact)
    ->
    (seed (* 10000 (time)))
  )

  ; deal and record first card (must be unique)

  (defrule generate-first-card
    (deck (cards $?D))
    ?which <- (deal (cardNum 0))
    ->
    (bind ?X (random 1 52))
    (printout t "X= " ?X " or " (nth$ ?X $?D) crlf)
    (assert (dealt (card-value ?X) (when last)))
    (modify ?which (cardNum 1))
  )
  ; remainder of deal

  (defrule generate-rem-cards
    (deck (cards $?D))
    ; which <- (deal (cardNum 0))
    ; test (< ?H 8) ; only 8 (unique) cards needed
    ; Last <- (dealt (when last))
    ->
    (modify ?Last (when ?H)) ; do this first
    (bind ?Y (random 1 52))
    (printout t "Y= " ?Y " or " (nth$ ?Y $?D) crlf)
    (assert (dealt (card-value ?Y) (when last))) ; may not be unique
    (modify ?which (cardNum (+ 1 ?H)))
  )
  ; check for duplicates -- see if any card matches the most recently dealt (last)
  ; if so, remove last card dealt and try again

  (defrule filter-duplicates
    (declare (salience 500))
    ?which <- (deal (cardNum ?H))
    ?prevlast <- (dealt (when = (- ?H 1))) ; return value constraint
    ; finds next to last
    ?last <- (dealt (card-value ?H) (when last)) ; latest
    ?dealt <- (dealt (card-value ?H) (when 'last)) ; not latest
    ->
    (printout t "duplicate detected -- removing" crlf)
    (retract ?last)
    (modify ?prevlast (when last))
    (modify ?which (cardNum (- ?H 1))) ; try again
  )
  ; form the hand (lots of variations on this)

  (defrule formHand
    (deal (cardNum 9)) ; done generating cards
    (dealt (card-value ?C1) (when 1))
    (dealt (card-value ?C2) (when 2))
    (dealt (card-value ?C3) (when 3))
    (dealt (card-value ?C4) (when 4))
    (dealt (card-value ?C5) (when 5))
    (dealt (card-value ?C6) (when 6))
    (dealt (card-value ?C7) (when 7))
    (dealt (card-value ?C8) (when 8))
    (dealt (card-value ?C9) (when last))
    ->
    (printout t "forming the hands" crlf)
    (assert (hand1 (holecard1 ?C1)
                  (holecard2 ?C2)
                  (community1 ?C5)
                  (community2 ?C6)
                  (community3 ?C7)
                  (community4 ?C8)
                  (community5 ?C9)))
    (assert (hand2 (holecard1 ?C3)
                  (holecard2 ?C4)
                  (community1 ?C5)
                  (community2 ?C6)
                  (community3 ?C7)
                  (community4 ?C8)
                  (community5 ?C9)))
  )
  ; non-ordered fact represents the deck

  (deffacts deckfact "the deck data structure"
    (deck (cards
      '2S' '3S' '4S' '5S' '6S' '7S' '8S' '9S' '10S' 'JS' 'QS' 'KS' 'AS'
      '2C' '3C' '4C' '5C' '6C' '7C' '8C' '9C' '10C' 'JC' 'QC' 'KC' 'AC'
      '2D' '3D' '4D' '5D' '6D' '7D' '8D' '9D' '10D' 'JD' 'QD' 'KD' 'AD'
      '2H' '3H' '4H' '5H' '6H' '7H' '8H' '9H' '10H' 'JH' 'QH' 'KH' 'AH')))

  (deffacts noDealYet (deal (cardNum 0))) ; no cards dealt
)

```

**Figure 7.9** Clips Example of Dealing Two Card Hands for Texas Holdem Card Game—Prelude to Expert System Development

```

CLIPS>(load "cardsOf.clips")
Defining deftemplate: deck
Defining deftemplate: dealt
Defining deftemplate: deal
Defining deftemplate: hand1
Defining deftemplate: hand2
Defining defrule: reseed +j
Defining defrule: generate-first-card +j+j
Defining defrule: generate-rem-cards =j+j+j
Defining defrule: filter-duplicates +j+j+j+j
Defining defrule: formHand +j+j+j+j+j+j+j+j+j+j
Defining deffacts: deckfact
Defining deffacts: noDealYet
TRUE
CLIPS> (reset)
CLIPS> f-0      (initial-fact) %% wordwrapped for illustration
f-1      (deck (cards '2S' '3S' '4S' '5S' '6S' '7S' '8S' '9S' '10S' 'JS' 'QS' 'KS' 'AS' '2C'
  '3C' '4C' '5C' '6C' '7C' '8C' '9C' '10C' 'JC' 'QC' 'KC' 'AC' '2D' '3D' '4D' '5D' '6D' '7D'
  '8D' '9D' '10D' 'JD' 'QD' 'KD' 'AD' '2H' '3H' '4H' '5H' '6H' '7H' '8H' '9H' '10H' 'JH' 'QH'
  'KH' 'AH'))
f-2      (deal (cardNum 0))
For a total of 3 facts.
CLIPS> X= 28 or '3D'
Y= 17 or '5C'
Y= 8 or '9S'
Y= 7 or '8S'
Y= 13 or 'AS'
Y= 11 or 'QS'
Y= 1 or '2S'
Y= 34 or '9D'
Y= 10 or 'JS'
forming the hands
CLIPS> f-0      (initial-fact)
f-1      (deck (cards '2S' '3S' '4S' '5S' '6S' '7S' '8S' '9S' '10S' 'JS' 'QS' 'KS' 'AS' '2C'
  '3C' '4C' '5C' '6C' '7C' '8C' '9C' '10C' 'JC' 'QC' 'KC' 'AC' '2D' '3D' '4D' '5D' '6D' '7D'
  '8D' '9D' '10D' 'JD' 'QD' 'KD' 'AD' '2H' '3H' '4H' '5H' '6H' '7H' '8H' '9H' '10H' 'JH' 'QH'
  'KH' 'AH'))
f-5      (dealt (card-value 28) (when 1))
f-8      (dealt (card-value 17) (when 2))
f-11     (dealt (card-value 8) (when 3))
f-14     (dealt (card-value 7) (when 4))
f-17     (dealt (card-value 13) (when 5))
f-20     (dealt (card-value 11) (when 6))
f-23     (dealt (card-value 1) (when 7))
f-26     (dealt (card-value 34) (when 8))
f-27     (dealt (card-value 10) (when last))
f-28     (deal (cardNum 9))
f-29     (hand1 (holecard1 28) (holecard2 17) (community1 13) (community2 11) (community3 1) (community4 34) (community5 10))
f-30     (hand1 (holecard1 8) (holecard2 7) (community1 13) (community2 11) (community3 1) (community4 34) (community5 10))
For a total of 14 facts.
CLIPS>

```

**Figure 7.10** Log of CLIPS Session Corresponding to Figure 7.9.

### 7.3.2 The agenda

CLIPS refers to the conflict set (`cs`) as the **agenda**. The **agenda** is a list of rules ranked in descending order of firing preference. In other words, the **agenda** is the list of all rules that have their LHS conditions satisfied (and have not yet been executed). In CLIPS, these rules are said to be *activated*.

When changes to working memory cause a rule to no longer be in the conflict set, it is said to be *deactivated*. One feature of CLIPS that makes a discussion of conflict resolution more complicated is the (optional) incorporation of adjustable rule *salience*, as shown in Section 7.3.3. The CLIPS user may view the status of rules using the function `matches` with syntax `matches <name-of-rule>`.

### 7.3.3 User-Defined Rule Salience

Placement of rules on the agenda is determined by *salience* or *saliency*. A noteworthy feature of CLIPS is that the user may define the saliency, or priority, of individual rules. This feature allows user specification of the relative importance of rules independent of any conflict resolution scheme. Salience may be dynamically controlled.

The rule with the highest salience (priority) will fire first. The declared salience value should be an expression that evaluates to an integer in the range  $-10,000$  to  $+10,000$ . The default salience value for a rule is zero. Also, by default, salience values are only evaluated when a rule is defined.

**Salience Example.** The CLIPS source is shown in the following example with three rules listed in order of increasing saliency. As the execution shows, saliency is used to prioritize the rule firings.

```
;; clips sample database for salience
;; template for memory

(deftemplate primitive
(slot name))

;; rules

(defrule rule1
(declare (salience -1000)) ;; not very important
(primitive (name a))
(primitive (name r))
=>
(assert (primitive (name b)))))

(defrule rule2
(declare (salience 1000)) ;; important
(primitive (name a))
(primitive (name r))
=>
(assert (primitive (name d)))))

(defrule rule3
(declare (salience 9999)) ;; very important
(primitive (name a))
(primitive (name r))
=>
(assert (primitive (name e)))))

;; initial facts

(deffacts startup
```



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

```

f-0      (initial-fact)
f-1      (primitive (name a))
f-2      (primitive (name r))
f-3      (primitive (name e))
f-4      (primitive (name d))
f-5      (primitive (name b))
For a total of 6 facts.
CLIPS> (agenda)
CLIPS>

```

### 7.3.4 Forming the Conflict Set (agenda)

A rule is said to be *activated* if the current state of `wm` satisfies all the CEs to the rule. In normal operation of the CLIPS execution cycle described in Section 7.1.4, rules may become (or remain) activated or become (or stay) deactivated. A newly activated rule is one that has just transitioned from a deactivated to an activated state. Most often, this is due to the addition (i.e., assertion) of one or more `wm` elements.

When a rule is newly activated, its placement on the agenda is based (in order) on the following factors:

1. Newly activated rules are placed above all rules of lower salience and below all rules of higher salience.
2. Among rules of equal salience, the current conflict resolution strategy is used to determine the placement among the other rules of equal salience.
3. If a rule is activated (along with several other rules) by the same assertion or retraction of a fact, and the preceding two tests are unable to specify an ordering, then the rule is arbitrarily (not randomly) ordered in relation to the other rules with which it was activated.

The default `depth` conflict resolution strategy implements a depth-first search by placing newly activated rule instantiations at the top of the agenda. Note that if a new fact in `wm` activates more than one instantiation of a rule at a time, the order of those instantiations (at the top of the agenda) is arbitrary.

### 7.3.5 Available CLIPS Conflict Resolution Strategies

CLIPS provides seven user-selectable conflict resolution strategies: `depth`, `breadth`, `simplicity`, `complexity`, `lex`, `mea`, and `random`. **The default conflict resolution strategy is `depth`.** A strategy is selected using the `set-strategy` command, with syntax:

```
(set-strategy <strategy>)
```

where `<strategy>`  $\in \{\text{depth}, \text{breadth}, \text{simplicity}, \text{complexity}, \text{lex}, \text{mea}, \text{random}\}$ . Examples are shown as follows. When the conflict resolution strategy is changed, the `agenda` is reordered. `(get-strategy)` returns the conflict resolution strategy currently in use. The descriptions of each apply to *productions of equal salience*. Refer to the CLIPS manual (`bpg.pdf`) for more detail.

A brief description of each follows:

**depth:** The depth strategy ranks productions based upon the recency of rule activation. Recency of activation is determined by the highest-valued time tag of the activated production. The most recently activated productions are ranked above all others. If a tie occurs, the agenda ordering of the most recently activated rules is arbitrary. As noted, this is the default strategy in CLIPS.

**breadth:** The breadth strategy is implemented by placing the newest activations of rules on the bottom of the agenda. Thus, the least recently activated productions are ranked above all others. As in depth, the order of any “ties” is arbitrary.

**simplicity/complexity:** This agenda ranking is based upon specificity or generality of a production. Specificity is measured in terms of LHS comparisons. For simplicity, the more comparisons, the lower the rule is on the agenda. The complexity ranking is the reverse, i.e., the more specific productions are placed on the agenda above those of lower specificity.

**lex:** This conflict resolution strategy emphasizes *recency*, but employs a more sophisticated strategy than **depth**. The **lex** strategy orders productions based upon sorting the recency of CE instantiation time tags. If necessary, a secondary test is to compare the number of CE tests (a crude measure of specificity).

**mea:** The basis of this strategy is forming the agenda by sorting (in decreasing order) activated productions using the time tag of their *first* CE.

**random:** Each activated production is assigned a random number to determine the placement of this production on the agenda.

### 7.3.6 Conflict Resolution Examples

We use two simple CLIPS databases to show the effect of varying CLIPS conflict resolution strategies. Figure 7.11 illustrates the simpler CLIPS conflict resolution strategies. The CLIPS source file is shown on the left. The resulting agendas, each formed by a different conflict resolution strategy, are shown on the right. Each reordered agenda is shown after revision of the conflict resolution strategy using **set-strategy**.

The reader should verify the reasonableness of these results, given the conflict resolution strategy descriptions. Figure 7.12 compares the **depth**, **lex**, and **mea** strategies.

## 7.4 Another Expert System Application Example: “Tenure”

### 7.4.1 An Application Domain

Consider the development of an “expert” system in CLIPS to automate the role of a college dean. The dean, among other things, is charged with determining which faculty members in his/her college receive tenure, or lifetime employment.

```

;; clips database for simple
;; conflict resolution examples
;; file: "ars-clips-cr.clips"
;; template for memory

(deftemplate primitive
  (slot name))

;; rules

(defrule rule1
  (primitive (name a))
  (primitive (name r))
  (primitive (name s))
=>
  (assert (primitive (name b)))))

(defrule rule2
  (primitive (name a))
  (primitive (name r))
=>
  (assert (primitive (name d)))))

(defrule rule3
  (primitive (name r))
  (primitive (name s))
=>
  (assert (primitive (name e)))))

(defrule rule4
  (primitive (name r))
  (primitive (name s))
  (test (< 1 2))
  (test (< 2 3))
=>
  (assert (primitive (name f)))))

;; initial facts

(deffacts startup
  (primitive (name a))
  (primitive (name r))
  (primitive (name s)))

```

CLIPS> (clear)

```

CLIPS> (load "ars-clips-cr.clips")
Defining deftemplate: primitive
Defining defrule: rule1 +j+j+j
Defining defrule: rule2 =j=j
Defining defrule: rule3 +j+j
Defining defrule: rule4 =j+j
Defining deffacts: startup
TRUE
CLIPS> (agenda)           ;; empty (no facts)
CLIPS> (get-strategy)
depth                   ;; the default
CLIPS> (reset)           ;; enable facts
CLIPS> (agenda)
0          rule1: f-1,f-2,f-3
0          rule3: f-2,f-3
0          rule4: f-2,f-3
0          rule2: f-1,f-2
For a total of 4 activations.
CLIPS> (set-strategy breadth)
depth ; note: returns previous strategy
CLIPS> (agenda)
0          rule2: f-1,f-2
0          rule4: f-2,f-3
0          rule3: f-2,f-3
0          rule1: f-1,f-2,f-3
For a total of 4 activations.
CLIPS> (set-strategy simplicity)
breadth
CLIPS> (agenda)
0          rule2: f-1,f-2
0          rule3: f-2,f-3
0          rule4: f-2,f-3
0          rule1: f-1,f-2,f-3
For a total of 4 activations.
CLIPS> (set-strategy complexity)
simplicity
CLIPS> (agenda)
0          rule4: f-2,f-3 ; more CE tests
0          rule1: f-1,f-2,f-3
0          rule2: f-1,f-2
0          rule3: f-2,f-3
For a total of 4 activations.
CLIPS> (set-strategy random)
complexity
CLIPS> (agenda)
0          rule1: f-1,f-2,f-3
0          rule4: f-2,f-3
0          rule2: f-1,f-2
0          rule3: f-2,f-3
For a total of 4 activations.
CLIPS>
```

Figure 7.11 CLIPS Database (left) to Illustrate Simple Conflict Resolution Strategies (depth, breadth, complexity, simplicity, and random). Right: Resulting Agenda Positions of Rules Using These Strategies.

```

;; clips database to illustrate
;; strategies depth, lex and mea.
;; file "ars-clips-cr2a.clips"

;; template for memory

(deftemplate primitive
(slot name))

;; rules

(defrule rule1
(primitive (name f1))
(primitive (name f4))
(primitive (name f5))
=>
(assert (primitive (name f6)))))

(defrule rule2
(primitive (name f2))
(primitive (name f4))
(primitive (name f5))
=>
(assert (primitive (name f7)))))

(defrule rule3
(primitive (name f1))
(primitive (name f4))
=>
(assert (primitive (name f8)))))

;; initial facts

(deffacts startup
(primitive (name f1))
(primitive (name f2))
(primitive (name f3))
(primitive (name f4))
(primitive (name f5)))

```

CLIPS> (get-strategy)  
depth  
CLIPS> (watch activations)  
CLIPS> (load "ars-clips-cr2a.clips")  
Defining deftemplate: primitive  
Defining defrule: rule1 +j+j+j  
Defining defrule: rule2 +j+j+j  
Defining defrule: rule3 =j=j  
Defining deffacts: startup  
TRUE  
CLIPS> (reset)  
==> Activation 0 rule3: f-1,f-4  
==> Activation 0 rule2: f-2,f-4,f-5  
==> Activation 0 rule1: f-1,f-4,f-5  
CLIPS> (facts)  
f-0 (initial-fact)  
f-1 (primitive (name f1))  
f-2 (primitive (name f2))  
f-3 (primitive (name f3))  
f-4 (primitive (name f4))  
f-5 (primitive (name f5))  
For a total of 6 facts.  
CLIPS> (agenda)  
0 rule1: f-1,f-4,f-5  
0 rule2: f-2,f-4,f-5  
0 rule3: f-1,f-4  
For a total of 3 activations.  
CLIPS> (set-strategy lex)  
depth  
CLIPS> (agenda)  
0 rule2: f-2,f-4,f-5  
0 rule1: f-1,f-4,f-5  
0 rule3: f-1,f-4  
For a total of 3 activations.  
CLIPS> (set-strategy mea)  
lex  
CLIPS> (agenda)  
0 rule2: f-2,f-4,f-5  
0 rule1: f-1,f-4,f-5  
0 rule3: f-1,f-4  
For a total of 3 activations.

Figure 7.12 Second CLIPS Conflict Resolution Examples. Here CLIPS conflict resolution strategies `depth`, `lex`, and `mea` are compared.

## 7.4.2 The Development Process

We begin the development of a CLIPS representation for this decision capability with the following dialog between a typical dean (DEAN) and the CLIPS software developer (CSD).

CSD: Dean, tell me how you decide who gets tenure.  
DEAN: That's easy. I award tenure to my faculty who publish,  
get research, and teach well.  
CSD: Am I correct in my understanding that they must do  
all three of these?  
DEAN: That's right.  
CSD: How does a faculty member "publish?"  
DEAN: The faculty member conducts research and documents  
the research.  
CSD: How does the faculty member get research?  
DEAN: The faculty member writes research proposals that  
subsequently become funded.  
CSD: What does it mean for a faculty member to teach well?  
DEAN: That's easy. A good teacher prepares lectures,  
delivers the lecture well, and gets good evaluations from the students.  
CSD: Is that all there is to it?  
DEAN: That's right.  
CSD: Thanks for your time and expertise, Dean.

## 7.4.3 Initial CLIPS Expert System Development Example

Based upon the conversation in Section 7.4.2, the following CLIPS representation was developed.

```
;; first 'tenure' example in clips
;; spartan version (no ranges, defaults, printouts,..)

;; TEMPLATES

(deftemplate faculty
  (slot name)
  (slot teaching_evals)
  (slot lecture_quality)
  (slot lecture_prep)
  (slot proposal_writing)
  (slot proposal_funding)
  (slot research_accomp)
  (slot research_publ)
  (slot tenure_status))

(deftemplate teaching_performance
  (slot name)
  (slot rating))
```

```

(deftemplate research_performance
  (slot name)
  (slot rating))

(deftemplate publ_performance
  (slot name)
  (slot rating))

;; RULES

(defrule assess_teach_perf
  (faculty (name ?W) (lecture_prep yes))
  (faculty (name ?W) (lecture_quality good))
  (faculty (name ?W) (teaching_evals good))
=>
  (assert (teaching_performance (name ?W) (rating acceptable)))))

(defrule assess_research_perf
  (faculty (name ?W) (proposal_writing extensive))
  (faculty (name ?W) (proposal_funding good))
=>
  (assert (research_performance (name ?W) (rating good)))))

(defrule assess_publ_perf
  (faculty (name ?W) (research_accomp substantial))
  (faculty (name ?W) (research_publ adequate))
=>
  (assert (publ_performance (name ?W) (rating good)))))

(defrule tenure-decision
  ?Who <- (faculty (name ?W) (tenure_status UNKNOWN))
  (teaching_performance (name ?W) (rating acceptable))
  (research_performance (name ?W) (rating good))
  (publ_performance (name ?W) (rating good))
=>
  (modify ?Who (tenure_status tenured)))

;; FACTS

(deffacts database
  (faculty (name pwd)
    (teaching_evals good)
    (lecture_quality good)
    (lecture_prep yes)
    (proposal_writing extensive)
    (proposal_funding good)
    (research_accomp substantial)
    (research_publ adequate)
    (tenure_status UNKNOWN))
  (faculty (name dir)
    (teaching_evals good)
    (lecture_quality fantastic)
    (lecture_prep yes)))

```

```
(proposal_writing extensive)
(proposal_funding good)
(research_accomp substantial)
(research_publ adequate)
(tenure_status UNKNOWN))

(faculty (name lls)
(teaching_evals good)
(lecture_quality good)
(lecture_prep never)
(proposal_writing extensive)
(proposal_funding good)
(research_accomp substantial)
(research_publ adequate)
(tenure_status UNKNOWN))

(faculty (name cwd)
(teaching_evals fair)
(lecture_quality good)
(lecture_prep yes)
(proposal_writing extensive)
(proposal_funding good)
(research_accomp extraordinary)
(research_publ adequate)
(tenure_status UNKNOWN))

(faculty (name lpr)
(teaching_evals good)
(lecture_quality good)
(lecture_prep yes)
(proposal_writing extensive)
(proposal_funding good)
(research_accomp substantial)
(research_publ adequate)
(tenure_status UNKNOWN)) )
```

Sample use with CLIPS is shown in Figure 7.13.

#### 7.4.4 Enhanced CLIPS Development of the Previous Example

Suppose the CLIPS software developer (CSD) from Section 7.4.2 later meets the Dean at the campus pub. The Dean, sitting in front of a table full of empty bottles, calls the CSD over and wishes to expand upon, and clarify, some of his earlier points. On this basis, the following, more comprehensive, database is developed. This expanded example illustrates the use of a number of more sophisticated CLIPS elements, including:

1. Non-ordered fact default values;
2. Restrictions on slot values;
3. (printout);
4. Field value constraint connectives (mostly |);
5. OR-ed condition elements;



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



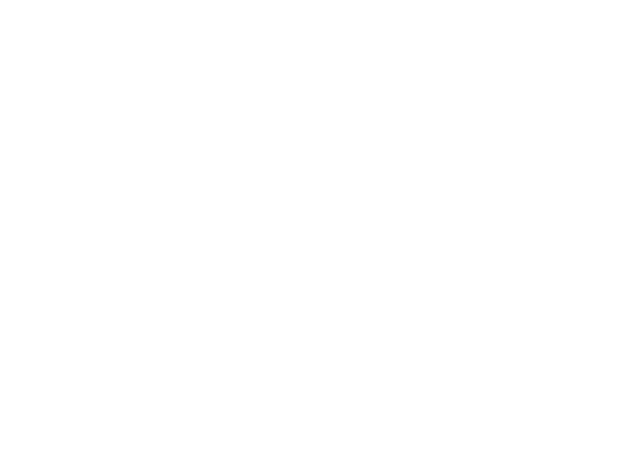
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

### COG Defuzzification

Let set  $U$  denote the set of possible fuzzy output values resulting from the application of system rules, i.e.,

$$U = \{U_1, U_2, \dots, U_n\} \quad (11.19)$$

The objective is to defuzzify the  $U_i$  values to achieve a crisp output, denoted  $U_{crisp}$ . *Center of Gravity Defuzzification* computes the output (crisp) value  $U_{out}^{COG}$  using:

$$U_{out}^{COG} = \frac{\sum_{i=1}^n \mu_i U_i}{\sum_{i=1}^n \mu_i} \quad (11.20)$$

COG defuzzification can lead to “prohibited control command” problems, and tends to produce a uniform distribution for the output set as the number of rules increases.

### MOM Defuzzification

The mean of maximum (MOM) technique is an alternative to the COG. Let  $G$  be the subset of  $U$  consisting of those values that are maxima of  $\mu$ , i.e.,

$$G = \{u^* | \mu(u^*) = \max_u \mu(u)\} \quad (11.21)$$

Let  $|G|$ , i.e., the cardinality of  $G$  be  $m$ . MOM computes the crisp output using:

$$U_{out}^{MOM} = \frac{\sum_{U_i \in G} U_i}{m} \quad (11.22)$$

[FY91] shows that COG and MOM defuzzification approaches have a common basis in terms of probability distributions.

### A Generalized Defuzzification, Basic Defuzzification Distribution (BADD) Method

This method, introduced by Filev and Yager [FY91], generalizes both the COG and MOM methods. The BADD technique can switch between the COG and MOM methods using a single parameter  $\alpha$ . The crisp output  $d$  is formulated as

$$d_G = \frac{\sum_{i=1}^n w_i^\alpha u_i}{\sum_{i=1}^n w_i^\alpha} \quad (11.23)$$

where the  $u_i$  are the elements of the discrete output universe and the  $w_i$  are the elements of the fuzzy output set. COG is achieved for  $\alpha = 1$ , and MOM is reached when  $\alpha \rightarrow \infty$ . The BADD defuzzification method can be used to influence the shape of the control surface from smooth ( $\alpha \approx 1$ ) to more step-shaped.

### Sugeno's CRI Method

The Sugeno model proposed by Takagi, Sugeno, and Kang [JS95] is an effort to develop a systematic approach to the generation of fuzzy rules. Typical Sugeno rules have a different form than the rule structure discussed in Section 11.3.3. Instead, they have the form:

$$\text{IF } [x] \text{ is } [A] \text{ and } [y] \text{ is } [B] \text{ THEN } [z] = f(x, y) \quad (11.24)$$

where  $A$  and  $B$  are fuzzy sets in the antecedent and  $z = f(x, y)$  is a crisp function. The output of each rule is then weighted by the rule's firing strength,  $\omega$ . The overall output  $d$  is formulated by

$$d = \frac{\sum_{i=1}^m w_i u_i}{\sum_{i=1}^m w_i} \quad (11.25)$$

where  $w_i$  are the elements of the fuzzy output set,  $u_i$  are the elements of the discrete output universe, and  $m$  is the number of fuzzy Sugeno rules. The method was not used in the vitrification research either because it requires the special rule structure.

### 11.5.2 Fuzzy Approach Shortcomings

Numerous issues are yet to be resolved regarding fuzzy system technology. They include:

- ◆ The low portability of rule bases. They tend to be application specific and thus lack standard libraries or modules for general applications;
- ◆ Significant sensitivity of control surface to small changes in the fuzzy rulebase or membership functions;
- ◆ Required time to identify and tune membership functions; and
- ◆ Validating a fuzzy system, especially for safety-critical applications.

## 11.6 Exploring Uncertainty and Fuzzy Concepts with CLIPS

### 11.6.1 Introduction to FuzzyCLIPS

FuzzyCLIPS is an enhanced version of CLIPS developed at the National Research Council of Canada. FuzzyCLIPS facilitates the implementation of fuzzy systems for reasoning with uncertainty. A good source of information is the users manual ("User's Guide"), which accompanies FuzzyCLIPS Version 6.10d.<sup>6</sup> FuzzyCLIPS is available without cost as a noncommercial download. The current URL is [http://www.iit.nrc.ca/IR\\_public/fuzzy/fuzzyClips/FuzzyCLIPSIndex2.html](http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyClips/FuzzyCLIPSIndex2.html) or <ftp://ai.iit.nrc.ca/pub/fzclips/>.

We show the use of FuzzyCLIPS for implementing a fuzzy production system that incorporates uncertainty in facts and rules. Specifically, both confidence factors and fuzzy sets are implemented in FuzzyCLIPS. Note that enhancing a production system like CLIPS to achieve this capability is a nontrivial venture. Several example applications are used.

---

<sup>6</sup>Written by Bob Orchard; dated October, 2004.

**Notes on FuzzyCLIPS and CLIPS COOL and Protégé.** FuzzyCLIPS implements the COOL functions described in Chapter 8, Section 8.3. This is important, since preexisting crisp (nonfuzzy) class-based representations may need to be “fuzzified,” or serve as the basis for a fuzzy representation. In addition, preexisting ontologies (with instances defined) developed in Protégé may also be used with FuzzyCLIPS. It is noteworthy, however, that instances of classes used with FuzzyCLIPS defined in Protégé (Chapter 8) must be explicitly declared “reactive,” i.e., able to be matched against object patterns in the CE to a rule. Otherwise, a FuzzyCLIPS error results.

### 11.6.2 Implementing Confidence Factors in FuzzyCLIPS

In FuzzyCLIPS, confidence factor-based uncertainty and fuzziness can be modeled simultaneously. First, we consider confidence factors. FuzzyCLIPS allows the use of confidence or “certainty” factors as follows:

```
(defrule flight-rule
  (declare (CF 0.95)) ;declares certainty factor of the rule
  (animal type bird)
  =>
  (assert (animal can fly))
)
```

This rule signifies that there is a 95% confidence in the rule stating that if an animal is a bird, then it can fly. In FuzzyCLIPS, if a certainty factor of a rule is not declared, it is assumed to be equal to 1.0. This is shown in the following example.

#### Default Behavior of FuzzyCLIPS

We begin by extending a previous CLIPS example containing no uncertainty. The CLIPS source is repeated here:

```
; clips implementation of a sample database
;; file: ars-clips.clips

;; template for memory
(deftemplate primitive
  (slot name))

;; a rule
(defrule rule1
  (primitive (name a))
  (primitive (name r))
  =>
  (assert (primitive (name b)))))

;; initial facts
(deffacts startup
  (primitive (name a))
  (primitive (name r))
  (primitive (name s)))
```

### **deftemplate** Extensions for Fuzzy Variables

In FuzzyCLIPS, all fuzzy variables must be defined before use with the **deftemplate** construct, which is an extension of the standard **deftemplate** construct in CLIPS. The extended syntax is:

```
(deftemplate <name> [<comments>]
    <from> <to> [<unit>] ; universe of discourse
    (
        t1
        .
        ; list of primary terms
        .
        tn
    )
)
```

where **<name>** is the identifier used for the fuzzy variable, **<from>** and **<to>** are floating point numbers, and a primary term **ti** ( $i=1, \dots, n$ ) has the form

```
(<pname> <description of fuzzy set>)
```

where **<pname>** represents the name of the the fuzzy set, and **<description of fuzzy set>** defines a corresponding membership function.

### Membership Functions

FuzzyCLIPS allows a membership function to be described using several techniques:

1. A so-called singleton representation, where points on the membership function are specified and interpolation is used in between these points;
2. A standard (including built-in) function representation; or
3. A linguistic expression that uses terms defined previously in a fuzzy **deftemplate** definition.

Built-in functions include the functions *S* (or *s*), *Z* (or *z*), and *PI* (or *pi*), as shown in Figure 11.8. A specification for these membership functions has the following syntax:

```
<standard> ::= (S a c) | (s a c) | (Z a c) | (z a c) | (PI d b) | (pi d b)
```

where **a**, **b**, **c**, and **d** are the parameters of the respective functions.

### Extension of **deffacts**

The **deffacts** construct has been expanded to allow the declaration of fuzzy facts with the syntax:

```
(deffacts <deffacts-name> [<comment>]
    <RHS-pattern>*
)
```

where **<RHS-pattern>** has been extended as follows:

```
<RHS-pattern> ::= <ordered-RHS-pattern> |
    <template-RHS-pattern> |
    <fuzzy-template-RHS-pattern>
```

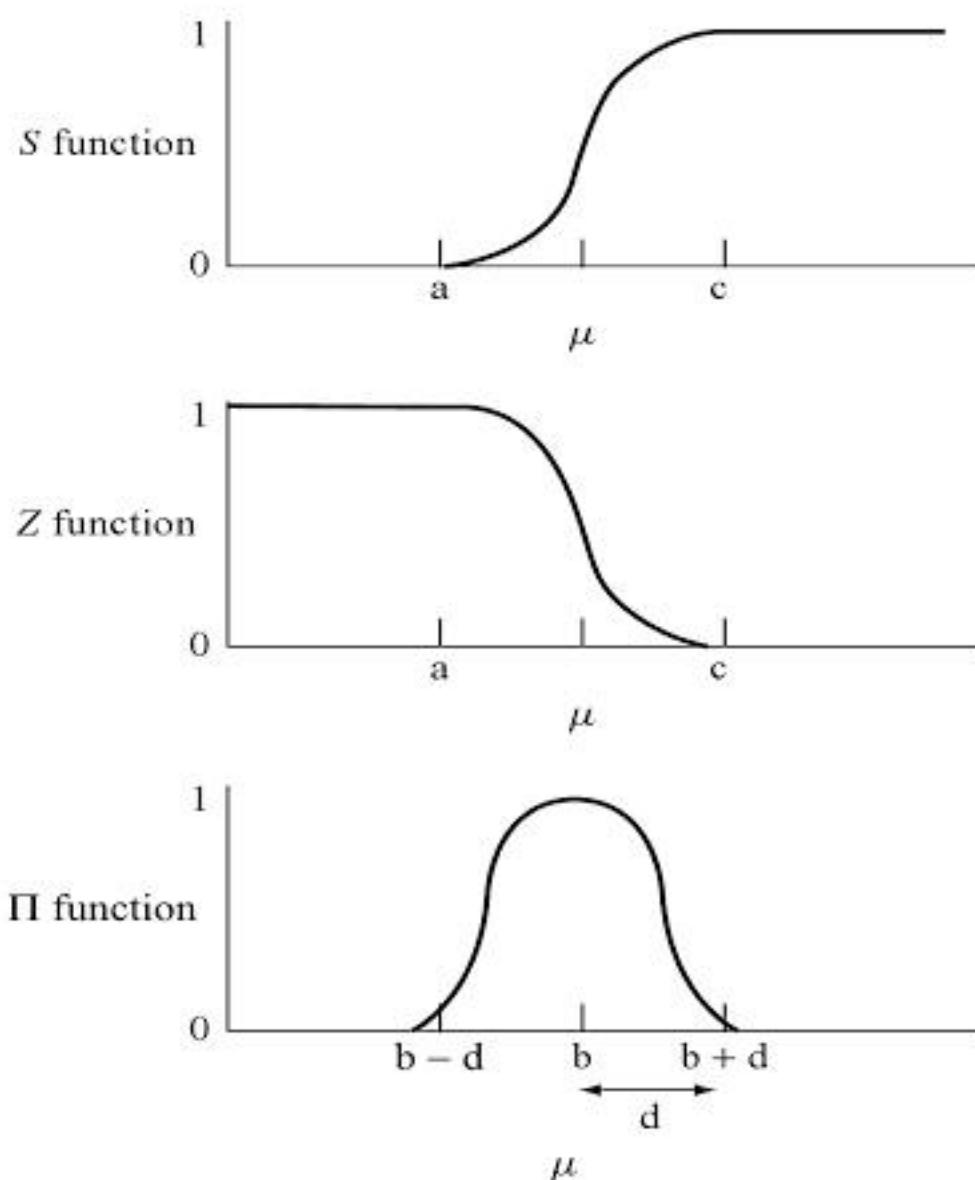


Figure 11.8 Built-in FuzzyCLIPS Membership Functions

```

<ordered-RHS-pattern> ::= (<symbol> <RHS-field>+)
    [CF <certainty factor> | <certainty factor expression>]

<template-RHS-pattern> ::= (deftemplate-name) <RHS-slot>*
    [CF <certainty factor> | <certainty factor expression>]

<fuzzy-template-RHS-pattern> ::=
    (<fuzzy-template-name> <description of fuzzy set>)
    [CF <certainty factor> | <certainty factor expression>]
  
```

### Example of Fuzzy Set Definition, Display, and Manipulation

We now show the use of a number of previously described constructs and topics via an example.

Consider the fuzzy representation of the concept of age. We begin with an example consisting of two principles:

1. “age” is a fuzzy variable that has membership in the fuzzy sets “young,” “middle,” and “old.”
2. An entity (in this example, a person) has a nonfuzzy slot with value “name” and a fuzzy slot representing the fuzzy variable age.

The FuzzyCLIPS representation for this is shown as follows:

```
;; examples of linguistic variable "age"

;; first using piecewise linear enumeration

(deftemplate age
  0 100 ; universe
  ( (young (0 1) (25 1) (40 0.5) (55 0))
    (middle (0 0) (25 0.5) (40 1.0) (55 0.5) (70 0))
    (old (0 0) (40 0) (55 0.5) (70 1) (80 1)) )
)

;; using 'standard function' representation
;; note: fuzzy sets may be different from above

(deftemplate sfage
  0 100 ; universe
  ( (young (z 30 55))
    (middle (pi 15 40))
    (old (s 40 70)) )
)

;; instances of entities with fuzzy attribute
;; first the fuzzy deftemplate

(deftemplate person
  (slot name)
  (slot age (type FUZZY-VALUE age))
)

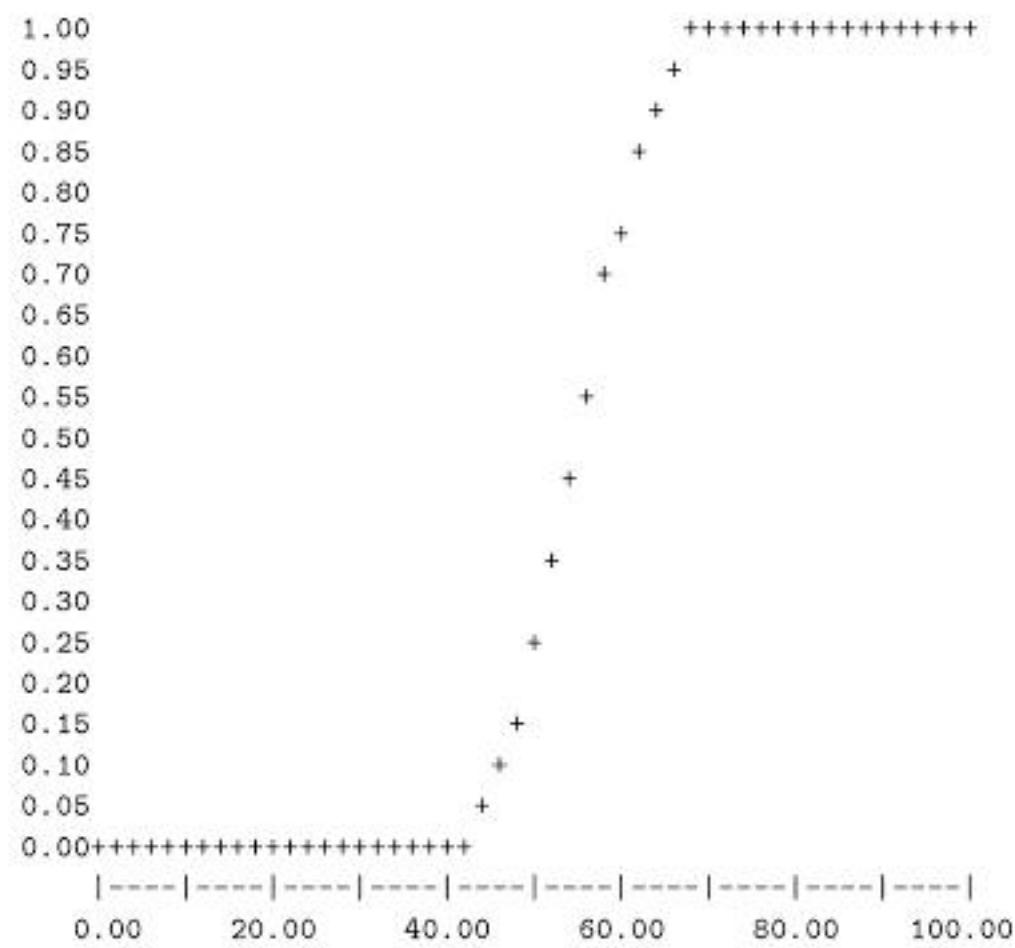
;; instances (fuzzy facts)

(deffacts startup
  (person (name bob) (age middle))
  (person (name katie) (age young)))
)
```

Before considering other aspects of the representation, we note that numerous manipulations with this fuzzy database in FuzzyCLIPS are possible. Some are shown here.

```
FuzzyCLIPS> (get-u age)
0.00 - 100.00
FuzzyCLIPS> (get-fuzzy-inference-type)
max-min
FuzzyCLIPS> (plot-fuzzy-value t + nil nil (create-fuzzy-value age middle)
)

Fuzzy Value: age
Linguistic Value: middle (+)
```



Universe of Discourse: From 0.00 to 100.00

FuzzyCLIPS>

### Fuzzy Set Operations

Here we consider the simple fuzzy set operations of union, intersection, and complement (not). First, consider the membership functions plotted in the following subsections for some simple fuzzy sets.

#### Fuzzy Sets for Low, Medium, and High Speed.

FuzzyCLIPS> (plot-fuzzy-value t + nil nil (create-fuzzy-value speed low))

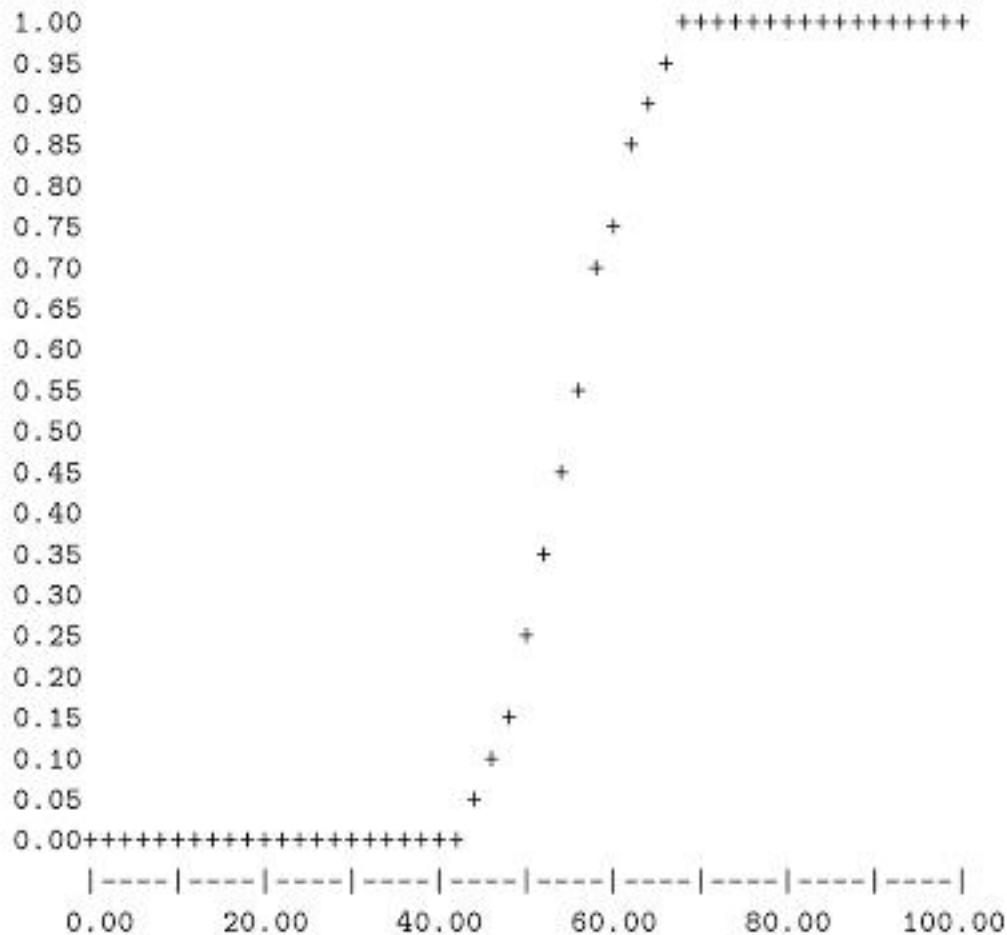
Fuzzy Value: speed  
Linguistic Value: low (+)

Universe of Discourse: From 0.00 to 100.00

FuzzyCLIPS> (plot-fuzzy-value t + nil nil (create-fuzzy-value speed high))

Fuzzy Value: speed

Linguistic Value: high (+)



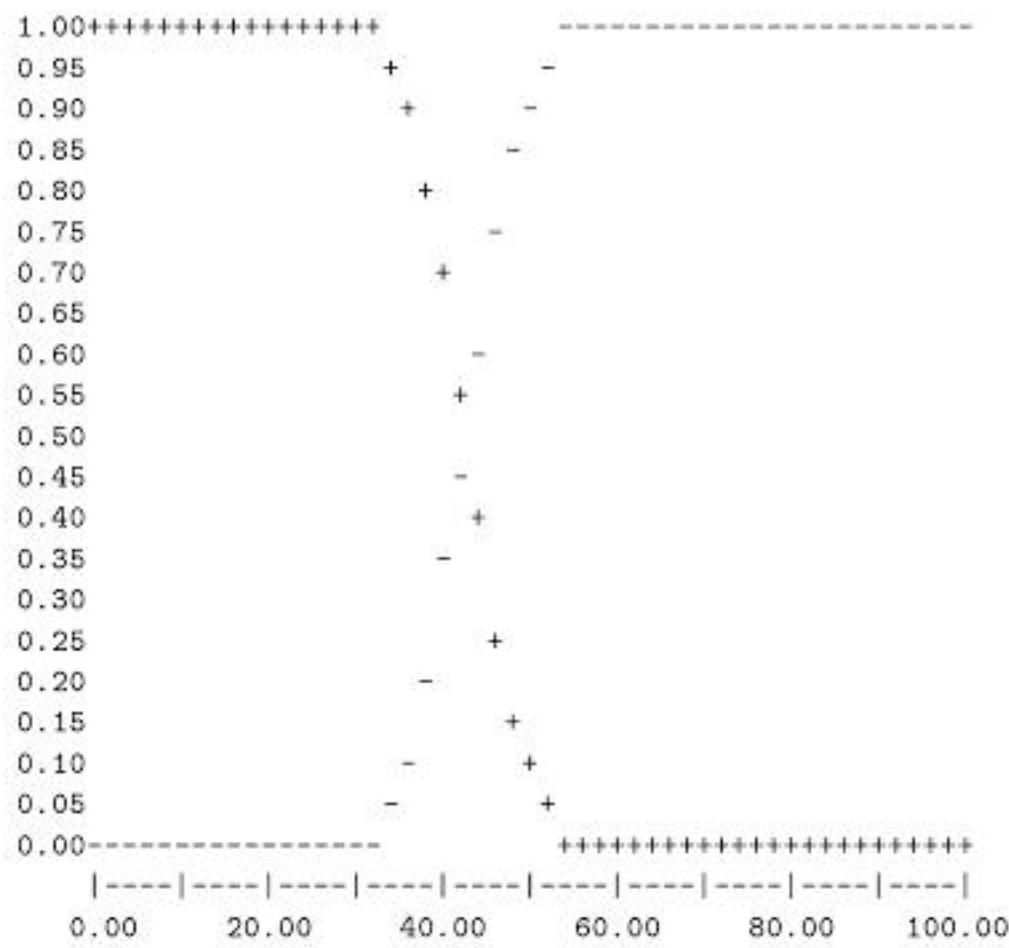
Universe of Discourse: From 0.00 to 100.00

### Example of Fuzzy Set Union.

```
FuzzyCLIPS> (plot-fuzzy-value t "abc" nil nil
  (create-fuzzy-value speed low) ;; 1st set to plot
  (create-fuzzy-value speed high) ;; 2nd set to plot
  (fuzzy-union (create-fuzzy-value speed low)
    (create-fuzzy-value speed high)) ;; 3rd set
)
```

Fuzzy Value: speed

Linguistic Value: low (a), high (b), [ low ] OR [ high ] (c)



Universe of Discourse: From 0.00 to 100.00

## Defuzzification

COG and MOM defuzzification is provided. This is shown in the examples that follow.

```
;; defuzzification of linguistic variable 'age'

;; first using piecewise linear enumeration

(deftemplate age
  0 100 ; universe
  ( (young (0 1) (25 1) (40 0.5) (55 0))
    (middle (0 0) (25 0.5) (40 1.0) (55 0.5) (70 0))
    (old (0 0) (40 0) (55 0.5) (70 1) (80 1)) )
)

;; using 'standard function' representation
;; note: fuzzy sets may be different from above

(deftemplate sfage
  0 100 ; universe
  ( (young (z 30 55))
    (middle (pi 15 40))
    (old (s 40 70)) )
)

;; instances of entities with fuzzy attribute
;; first the fuzzy deftemplate
```

```

(deftemplate person
  (slot name)
  (slot age (type FUZZY-VALUE age))
)

;; sample rule for COG-defuzzification

(defrule age-defuzz-cog
  ?any <- (person (name ?Who) (age ?FSet))
=>
  (printout t "person: " ?Who crlf
           "has COG-defuzzified age= " (moment-defuzzify ?FSet)
           crlf crlf)
)

;; sample rule for MOM-defuzzification

(defrule age-defuzz-mom
  ?any <- (person (name ?Who) (age ?FSet))
=>
  (printout t "person: " ?Who crlf
           "has MOM-defuzzified age= " (maximum-defuzzify ?FSet)
           crlf crlf)
)

;; instances (fuzzy facts)

(deffacts startup
  (person (name bob) (age middle))
  (person (name katie) (age young)))
)

FuzzyCLIPS> (load "age-defuzz.fclips")
Defining deftemplate: age
Defining deftemplate: sfage
Defining deftemplate: person
Defining defrule: age-defuzz-cog +j
Defining defrule: age-defuzz-mom +j
Defining deffacts: startup
TRUE
FuzzyCLIPS> (reset)
FuzzyCLIPS> (facts)
f-0      (initial-fact) CF 1.00
f-1      (person (name bob) (age middle)) CF 1.00
  ( (0.0 0.0) (25.0 0.5) (40.0 1.0) (55.0 0.5) (70.0 0.0) )
f-2      (person (name katie) (age young)) CF 1.00
  ( (25.0 1.0) (40.0 0.5) (55.0 0.0) )
For a total of 3 facts.
FuzzyCLIPS> (run)
person: katie
has COG-defuzzified age= 20.9375

person: katie
has MOM-defuzzified age= 12.5

```



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



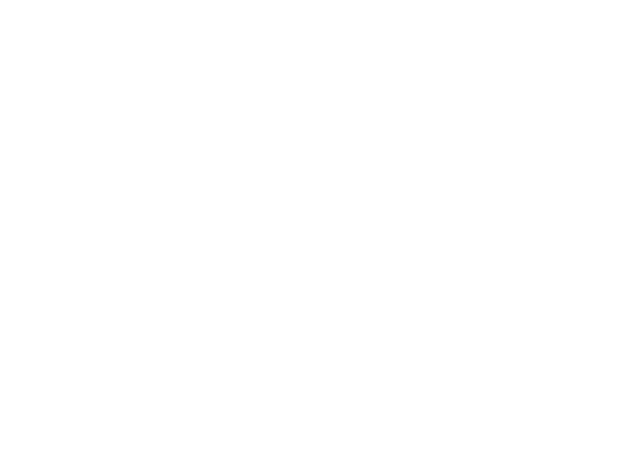
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



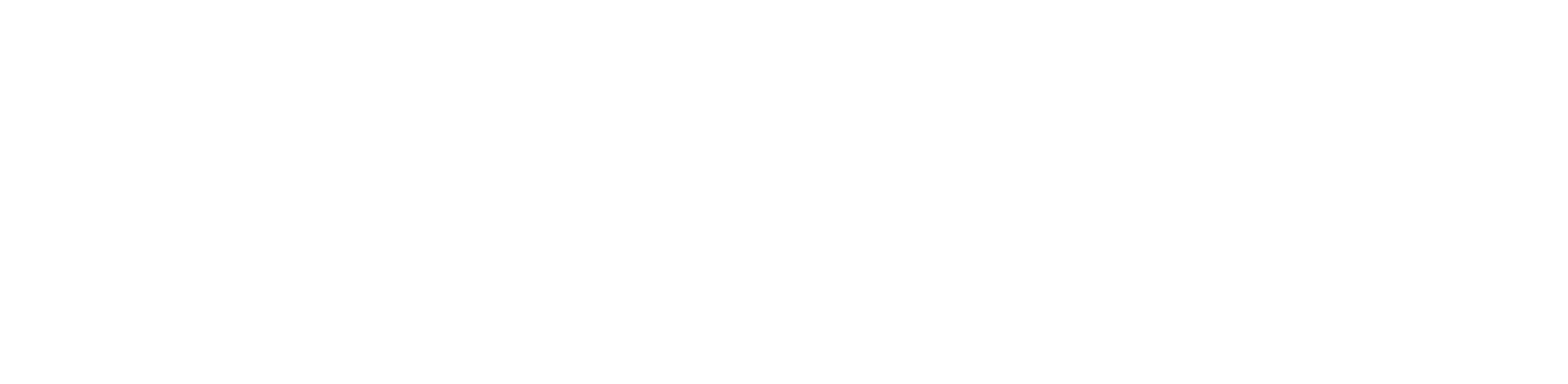
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



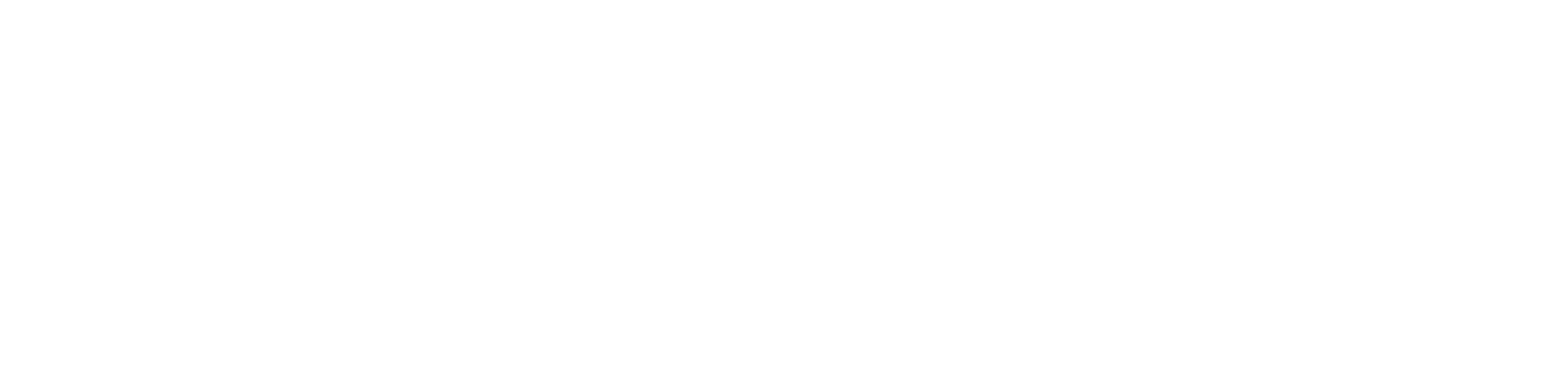
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



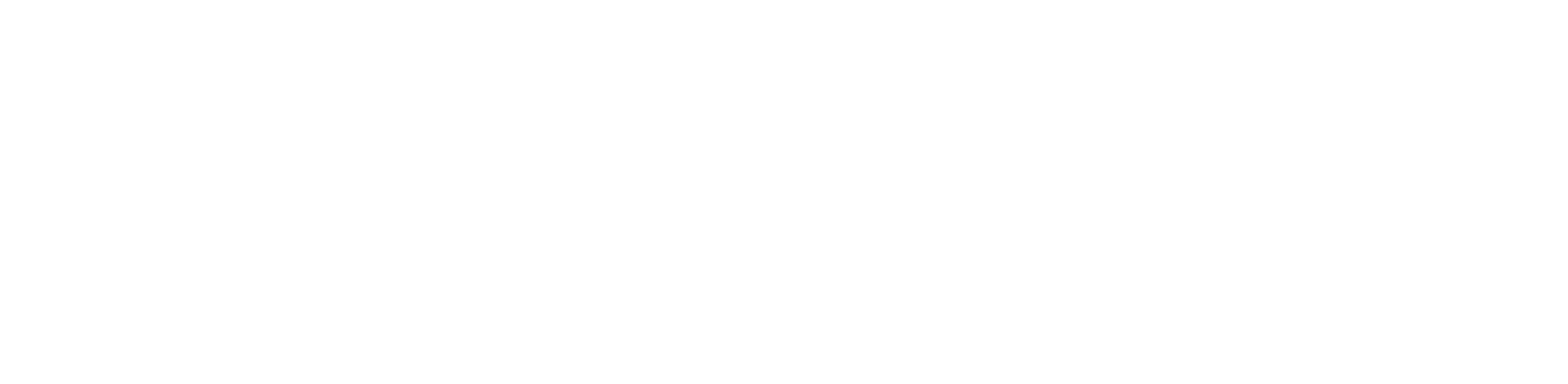
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



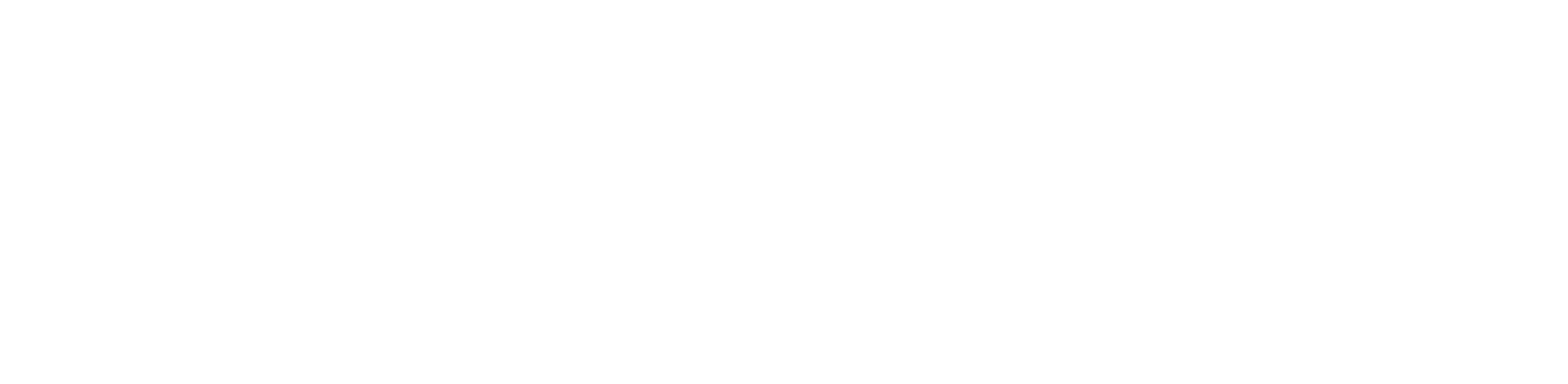
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



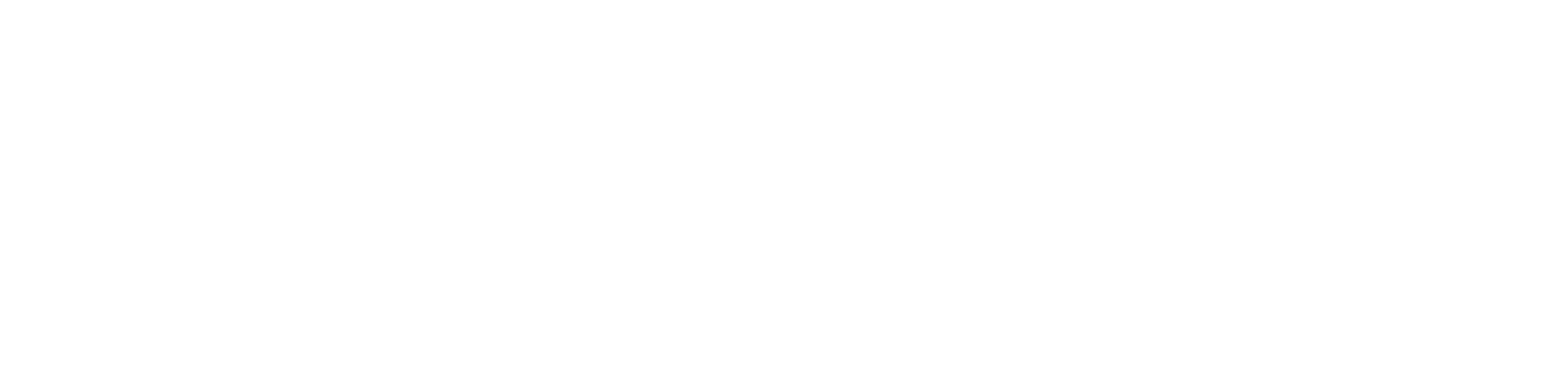
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



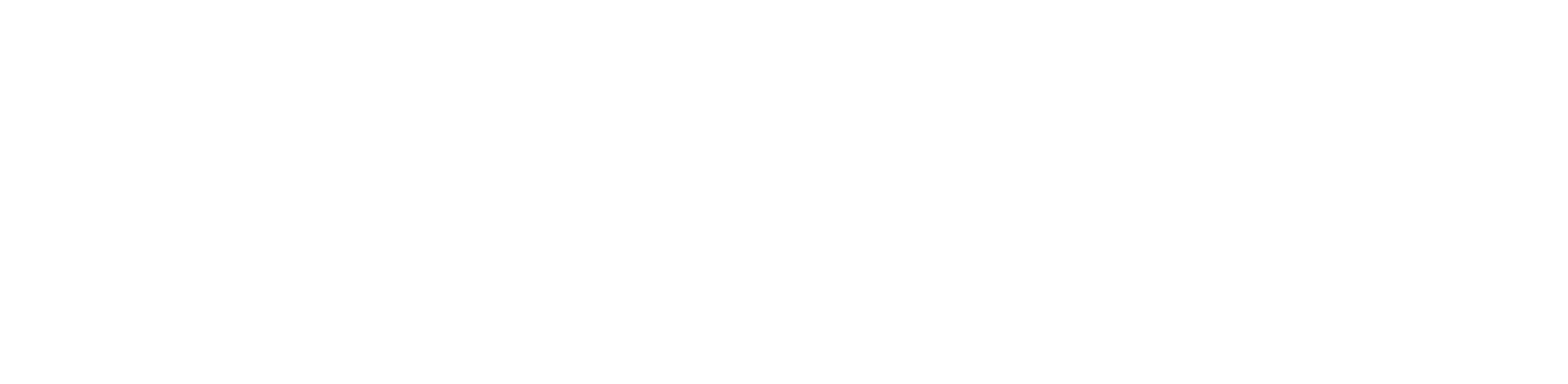
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

# Index

- c-means algorithm, 571
- A\* algorithm, 67
- activation function, 495, 503, 514
- adaptation, 649, 653
- agenda, 182, 195, 210
- agenda, 196, 210, 212, 215
- agents, 253, 295, 321, 329–345, 417, 694
- Alan Turing, 5
- anonymous variable, 611, 714
- antecedent, 179–192, 210, 229, 272, 356
- artificial ant colony algorithm, 697
- artificial genetic algorithm, 650–651
- artificial neural networks, 2, 493, 551, 649, 679
- associative memory, 530
- attribute grammar, 146
- attributes, 24, 203, 297, 572, 616
- axiom, 168
- back propagation, 508–511, 523
- backtracking, 99
- BAM recall, 557
- basic function groups (Lisp), 735
- Bayes Rule, 355, 358
- Belief Network (BN), 362
- Bidirectional Associative Memory (BAM), 551
- binary relation, 20, 707
- biological systems, 493, 649
- biomimetics, 649
- blocks world, 418, 425, 517
- blueprint representation, 684
- bottom-up processing, 4
- content addressable memory (CAM), 527, 533, 534, 536
- cardinality, 705
- case-based reasoning (CBR), 598, 599, 602
- chain rule, 503, 729
- chaining, 167, 169, 172, 178
- character association, 560
- chromosome, 650
- chunking, 20, 295
- class hierarchy, 258
- classification, 514, 640
- classification rules, 642
- clause, 74
- clause (Prolog), 146, 711
- CLIPS, 15–16, 195–242, 254, 362
- clustering, 569–572, 574, 591
- column vector, 554, 724
- competitive learning, 574
- complexity function, 48
- compositional rules of inference (CRI), 375, 382–383, 404
- computational complexity, 13, 45, 227, 680
- cond (Lisp), 736
- condition element (CE), 263, 302
- conditional probability, 354
- confidence factor, 353, 356, 360, 388–389
- conflict resolution, 181–183, 210, 215, 295, 305
- conflict set, 182, 196
- constrained optimization, 82, 541
- constraint propagation, 90, 100
- constraint satisfaction, 527, 540, 564
- constraint satisfaction problem (CSP), 65–121

- constructive inductive generalization, 612  
 COOL, 259–290  
 covering, 603  
 crossover, 651, 655, 658, 665, 684, 693  
 CYC, [33](#)  
 cycle, 422  
 data flow graph, 231  
 data structures (Lisp), 733  
 decision tree, 613–640  
 declarative, [20](#), 733  
 declarative programming, 711  
**defacts**, 204  
**deftemplate**, 203  
**defun**, 735  
 defuzzification, 380, 382, 385  
 depth-first search, 65, [66](#), 99, 714  
 description modification, 605  
 diagnosis, 316–317  
 digraph, [21](#), [52](#), 90, 708  
 dimensionality reduction, 574  
 discrete mathematics, [20](#), 167, 705  
 discrete relaxation, 90  
 distance measure, 487, 570, 727  
 ELIZA, 6, 133  
 entropy, 614, 616  
 equilibrium state, 530  
 evolution, 578, 696  
 evolutionary algorithms, 649  
 exemplars, 613, 640  
 expert system, 35, 121, 170, [216](#), 375  
 fact (Prolog), 711  
 forward chaining, 192  
 frame, 23  
 frame problem, 420  
 function, 707, 733  
 fuzzification, 380, 382  
 fuzzy antecedent, 381  
 fuzzy expert system, 375  
 fuzzy logic, 169, 353, 373  
 fuzzy production system, 375  
 fuzzy sets, 375  
 FuzzyCLIPS, [387](#)  
 game theory, 11, [68](#)  
**gcc**, 692  
 gene, 650  
 generalization, 156, 602, 632  
 generalization operators, 609–610  
 generalization with exceptions, 611  
 generalization-based learning, 602  
 Generalized Delta Rule (GDR), 501, 511  
 generate-and-test (GAT), 13, 64, 75, 427  
 genetic operator, 652  
 genome, 650  
 genotype, 650  
 goal (Prolog), 712  
 goal state, 309, 417  
 gradient descent, 501, 505, 729  
 grammars, 136  
 grammar types, 138  
 graph, 708  
 growing neural gas (GNG), 588  
 head (clause), 711  
 Hebbian learning, 532  
 heuristic, 59, 99, 417  
 hidden layers, 513  
 Hopfield network, 528  
 ID3 algorithm, 612–633  
 impasse, 295, 325  
 implication, 169  
 induction, 168, 598, 608, 615  
 inference, 168  
 inference engine (IE), 180  
 instances, 262  
 John Hopfield, 527  
 kernel, 295  
 knowledge base, 27  
 knowledge representation, [19](#)  
 labeling, 76–79 200, 314  
 learning from examples, 597–598

- learning, 1, 295, 327, 569, 595  
 learning algorithm, 576  
 learning as reuse, 596  
 learning by analogy, 597  
 learning from examples, 596  
 learning rate, 505, 520  
 Liapunov, 536  
 limit set, 91  
 linear algebra, 723  
 linguistic variable, 380  
 Lisp interpreter, 734  
 list (Prolog), 717  
 list manipulation, 735  
 local minima, 527  
 logic, 150, 167  
 logistic activation function, 496
- manipulation, 1, 167  
 map coloring problem, 71, 85  
 matrix, 723  
 matrix rank, 726  
 membership functions, 375  
 Modus Ponens (MP), 172, 274, 719  
 momentum, 513  
 multiagent or parallel planning, 436  
 multilayer feedforward structure, 497  
 multiple inheritance, 264  
 multivalued logic, 169, 353, 364  
 musical structures, 621  
 mutation, 652
- natural language processing (NLP), 11, 293  
 natural language understanding, 132  
 negation-as-failure, 719  
 neighborhood, 578  
 network capacity, 533  
 network energy function, 539  
 network topology, 112  
 neural gas (NG), 583–587  
 neurons, 494  
 neuroscience, 5  
 NG algorithm, 583  
 noncommutative production system, 419
- nonmonotonic logic, 273–274  
 object-oriented (OO) representation, 258  
 observation/explanation-based learning, 597  
 Octave, 94, 97, 511  
 ontology, 26–35, 265  
 Ontology Web Language (OWL), 28  
 operator, 47, 294, 307, 429  
 optimization, 60, 527, 540, 651, 684, 729  
 outer product, 532  
 overgeneralization, 611  
 partial recall, 553  
 partition, 706  
 path planning, 418  
 phenotype, 650  
 pheromone, 695  
 planning, 225, 417–488  
 planning algorithms, 424  
 positive and negative exemplars, 605  
 predicate, 711, 717  
 preference resolution, 306  
 probability, 353, 354, 360  
 procedural, 20  
 production, 262, 294, 330, 721  
 production memory, 298  
 production system, 42, 185, 195, 293, 322,  
     419, 421  
 programming paradigms, 15  
 Prolog, 75, 93, 140, 273, 489, 711  
 propositional satisfiability (SAT), 73  
 Protégé, 29, 265  
 psychology, 3, 597
- read-eval-print loop (EVAL), 734  
 recurrent networks, 527  
 recursion, 716  
 representation, 1, 19, 167, 293, 419, 421,  
     596, 651  
 reproduction, 651  
 resolution, 176  
 Resource Description Framework (RDF),  
     28  
 Rete algorithm, 171, 226, 229

- rule (Prolog), 711  
 rule-based inference, 417  
 rule-based system, 170, 178, 187  
 s-expression, 734  
 schema, 597  
 search, 13, [45](#), 51–57, 62, 65, 74, 99, 419, 422, 651, 653  
 search problem, [52](#)  
 selection strategy, 654  
 self-organizing feature map (SOFM), 571, 574  
 self-organizing network, 576  
 Semantic Computing, 11  
 semantic net, [21](#)  
 Semantic Web, 23  
 semantics, 144  
 sequential decision, 613  
 set, 705  
 similarity measure, 570, 601, 727  
 Soar, 293  
 soft computing, 373  
 sorting, 206, 584  
 specialization-based learning, 602  
 squashing, 495–496  
 stable state, 530  
 state, 294, 297, 419, 527, 529  
 state elaboration, 295, 304, 307  
 state energy, 530  
 state space, [45](#), 492  
 state-space graph, [52](#)  
 storage prescription, 531  
 STRIPS, 424, 426  
 subgoal, 714  
 swarm-based algorithm, 698  
 swarm intelligence, 694  
 synapse, 495  
 system state, 295  
 tail, 718  
 tail (clause), 711  
 top-down processing, 4  
 topological neighborhoods, 575  
 topologically ordered, 574  
 training, 501, 588  
 training set, 518, 534, 556, 603  
 Traveling Salesman Problem (TSP), 548, 680  
 tree induction, 612  
 triangle table, 428  
 Turing test, 14  
 uncertainty, 353, 362, 373  
 unification, 13, 172  
 unifier, 173  
 unit bias, 511, 547  
 unit net activation, 495  
 vector inner product, 726  
 vector outer product, 726  
 William James, 3  
 World Wide Web Consortium (W3C), 22  
 working memory, 296