



ARIZONA STATE UNIVERSITY  
GENERAL STUDIES COURSE PROPOSAL COVER FORM

**Course information:**

Copy and paste current course information from Class Search/Course Catalog.

Academic Unit	<u>Historical, Philosophical and Religious Studies</u>	Department	<u>Philosophy</u>
Subject	<u>PHI</u>	Number	<u>319</u>
Title	<u>Philosophy, Computing and Artificial Intelligence</u>		Units: <u>3</u>
Is this a cross-listed course? If yes, please identify course(s)	<u>No</u>		
Is this a shared course? Course description:	<u>Yes</u>	If so, list all academic units offering this course	<u>SLS, NCIAS</u>

**Requested designation:** Mathematical Studies-CS

Note- a separate proposal is required for each designation requested

**Eligibility:**

Permanent numbered courses must have completed the university's review and approval process.  
For the rules governing approval of omnibus courses, contact [Phyllis.Lucie@asu.edu](mailto:Phyllis.Lucie@asu.edu) or [Lauren.Leo@asu.edu](mailto:Lauren.Leo@asu.edu).

**Submission deadlines dates are as follow:**

For Fall 2015 Effective Date: October 9, 2014

For Spring 2016 Effective Date: March 19, 2015

**Area(s) proposed course will serve:**

A single course may be proposed for more than one core or awareness area. A course may satisfy a core area requirement and more than one awareness area requirements concurrently, but may not satisfy requirements in two core areas simultaneously, even if approved for those areas. With departmental consent, an approved General Studies course may be counted toward both the General Studies requirement and the major program of study.

**Checklists for general studies designations:**

Complete and attach the appropriate checklist

- [Literacy and Critical Inquiry core courses \(L\)](#)
- [Mathematics core courses \(MA\)](#)
- [Computer/statistics/quantitative applications core courses \(CS\)](#)
- [Humanities, Arts and Design core courses \(HU\)](#)
- [Social-Behavioral Sciences core courses \(SB\)](#)
- [Natural Sciences core courses \(SQ/SG\)](#)
- [Cultural Diversity in the United States courses \(C\)](#)
- [Global Awareness courses \(G\)](#)
- [Historical Awareness courses \(H\)](#)

**A complete proposal should include:**

- Signed General Studies Program Course Proposal Cover Form
- Criteria Checklist for the area
- Course Catalog description
- Course Syllabus
- Copy of Table of Contents from the textbook and list of required readings/books

**Respectfully request that proposals are submitted electronically with all files compiled into one PDF. If necessary, a hard copy of the proposal will be accepted.**

**Contact information:**

Name Cindy Baade Phone 5-7183  
Mail code 4302 E-mail: cynthia.baade@asu.edu

**Department Chair/Director approval: (Required)**

Chair/Director name (Typed): Matthew J. Garcia Date: 2/3/15  
Chair/Director (Signature):

Arizona State University Criteria Checklist for

**MATHEMATICAL STUDIES [CS]**

**Rationale and Objectives**

The **Mathematical Studies** requirement is intended to ensure that students have skill in basic mathematics, can use mathematical analysis in their chosen fields, and can understand how computers can make mathematical analysis more powerful and efficient. The **Mathematical Studies** requirement is completed by satisfying both the **Mathematics [MA]** requirement and the **Computer/Statistics/Quantitative Applications [CS]** requirement explained below.

The **Mathematics [MA]** requirement, which ensures the acquisition of essential skill in basic mathematics, requires the student to complete a course in College Mathematics, College Algebra, or Pre-calculus; or demonstrate a higher level of skill by completing a mathematics course for which a course in the above three categories is a prerequisite.

The **Computer/Statistics/Quantitative Applications [CS]** requirement, which ensures skill in real world problem solving and analysis, requires the student to complete a course that uses some combination of computers, statistics, and/or mathematics.\* Computer usage is encouraged but not required in statistics and quantitative applications courses. At a minimum, such courses should include multiple demonstrations of how computers can be used to perform the analyses more efficiently.

\*CS does *not* stand for computer science in this context; the "S" stands for statistics. Courses in computer science must meet the criteria stated for CS courses.

Revised April 2014

Proposer: Please complete the following section and attach appropriate documentation.

<b>ASU--[CS] CRITERIA</b>			
A COMPUTER/STATISTICS/QUANTITATIVE APPLICATIONS [CS] COURSE MUST SATISFY ONE OF THE FOLLOWING CRITERIA: 1, 2, OR 3			
YES	NO		Identify Documentation Submitted
		<b>1. Computer applications*:</b> courses must satisfy both a and b:	
<input checked="" type="checkbox"/>	<input type="checkbox"/>	a. Course involves the use of computer programming languages or software programs for quantitative analysis, algorithmic design, modeling, simulation, animation, or statistics.	syllabus, table of contents of required books
		b. Course requires students to analyze and implement procedures that are applicable to at least one of the following problem domains ( <b>check those applicable</b> ):	
<input type="checkbox"/>	<input type="checkbox"/>	i. Spreadsheet analysis, systems analysis and design, and decision support systems.	
<input type="checkbox"/>	<input type="checkbox"/>	ii. Graphic/artistic design using computers.	
<input type="checkbox"/>	<input type="checkbox"/>	iii. Music design using computer software.	
<input type="checkbox"/>	<input type="checkbox"/>	iv. Modeling, making extensive use of computer simulation.	
<input type="checkbox"/>	<input type="checkbox"/>	v. Statistics studies stressing the use of computer software.	
<input checked="" type="checkbox"/>	<input type="checkbox"/>	vi. Algorithmic design and computational thinking.	syllabus, table of contents of required books
<p>*The <b>computer applications</b> requirement <b>cannot</b> be satisfied by a course, the content of which is restricted primarily to word processing or report preparation skills, the study of the social impact of computers, or methodologies to select software packages for specific applications. Courses that emphasize the use of a computer software package are acceptable only if students are required to understand, at an appropriate level, the theoretical principles embodied in the operation of the software and are required to construct, test, and implement procedures that use the software to accomplish tasks in the applicable problem domains. Courses that involve the learning of a computer programming language are acceptable only if they also include a substantial introduction to applications to one of the listed problem domains.</p>			

YES	NO		Identify Documentation Submitted
<b>2. Statistical applications: courses must satisfy a, b, and c.</b>			
<input type="checkbox"/>	<input checked="" type="checkbox"/>	a. Course has a minimum mathematical prerequisite of College Mathematics, College Algebra, or Pre-calculus, or a course already approved as satisfying the MA requirement.	
		b. The course must be focused principally on developing knowledge in statistical inference and include coverage of all of the following:	
<input type="checkbox"/>	<input type="checkbox"/>	i. Design of a statistical study.	
<input type="checkbox"/>	<input type="checkbox"/>	ii. Summarization and interpretation of data.	
<input type="checkbox"/>	<input type="checkbox"/>	iii. Methods of sampling.	
<input type="checkbox"/>	<input type="checkbox"/>	iv. Standard probability models.	
<input type="checkbox"/>	<input type="checkbox"/>	v. Statistical estimation	
<input type="checkbox"/>	<input type="checkbox"/>	vi. Hypothesis testing.	
<input type="checkbox"/>	<input type="checkbox"/>	vii. Regression or correlation analysis.	
<input type="checkbox"/>	<input type="checkbox"/>	c. The course must include multiple demonstrations of how computers can be used to perform statistical analysis more efficiently, if use of computers to carry out the analysis is not required.	

YES	NO		Identify Documentation Submitted
		<b>3. Quantitative applications:</b> courses must satisfy a, b, and c:	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	a. Course has a minimum mathematical prerequisite of College Mathematics, College Algebra, or Pre-calculus, or a course already approved as satisfying the MA requirement.	
		b. The course must be focused principally on the use of mathematical models in quantitative analysis and decision making. Examples of such models are:	
<input type="checkbox"/>	<input type="checkbox"/>	i. Linear programming.	
<input type="checkbox"/>	<input type="checkbox"/>	ii. Goal programming.	
<input type="checkbox"/>	<input type="checkbox"/>	iii. Integer programming.	
<input type="checkbox"/>	<input type="checkbox"/>	iv. Inventory models.	
<input type="checkbox"/>	<input type="checkbox"/>	v. Decision theory.	
<input type="checkbox"/>	<input type="checkbox"/>	vi. Simulation and Monte Carlo methods.	
<input type="checkbox"/>	<input type="checkbox"/>	vii. Other (explanation must be attached).	
<input type="checkbox"/>	<input type="checkbox"/>	c. The course must include multiple demonstrations of how computers can be used to perform the above applications more efficiently, if use of computers is not required by students.	

Course Prefix	Number	Title	General Studies Designation
PHI	319	Philosophy, Computing, and Artificial Intelligence	CS

Explain in detail which student activities correspond to the specific designation criteria. Please use the following organizer to explain how the criteria are being met.

Criteria (from checksheet)	How course meets spirit (contextualize specific examples in next column)	Please provide detailed evidence of how course meets criteria (i.e., where in syllabus)
1a	This course involves the use of Prolog and logic programming to model certain aspects of intelligence within the context of an observation-thought-decision-action cycle	Units 1-7
1bvi	This course requires students to analyze and implement procedures applicable to algorithmic design and computational thinking. Students analyze and implement procedures for observation, forward reasoning, abduction, maintenance goal triggering, backward reasoning, achievement goal reduction, plan selection, and prohibition. Students analyze the differences between classical negation and negation-as-failure as well as the differences between conclusive and defeasible reasoning more generally. Students also analyze and implement procedures for lexicon building, recognizing grammatical sentences, model building, and determining the truth value of sentences relative to models.	Units 1-7

--	--	--

PHI 319 **Philosophy, Computing and Artificial Intelligence**

3 CS

Philosophical problems surrounding artificial intelligence (AI). Thinking as computation and AI; the ethics, epistemology and metaphysics of computing.

**Allow multiple enrollments:** No

**Primary course component:** Lecture

**Repeatable for credit:** No

**Grading method:** Standard Grading

**Offered by:**

New College of Interdisciplinary Arts and Sciences -- School of Humanities, Arts, and Cultural Studies

Pre-requisites: ENG 102, 105 or 108 with C or better; Minimum 25 hours

College of Letters and Sciences -- College of Letters and Sciences

Pre-requisites: ENG 102, 105 or 108 with C or better; Minimum 25 hours

College of Liberal Arts and Sciences -- Historical, Philosophical & Religious Studies, Sch

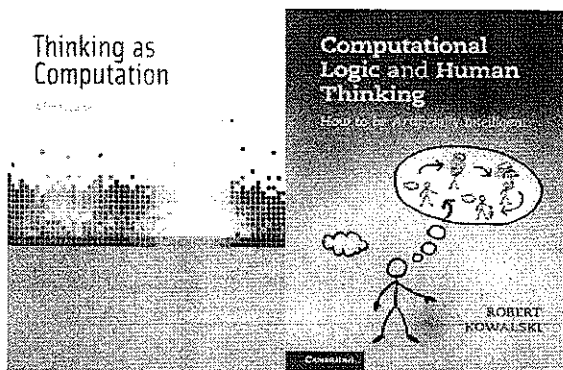
Pre-requisites: ENG 102, 105 or 108 with C or better; Minimum 25 hours



# Philosophy, Computing, and Artificial Intelligence

PHI 319 (Spring 2015, Session C, 15 Weeks; T,Th 12:00-1:15 Tempe ECG-G227)

Thomas A. Blackson  
School of Historical, Philosophical, and Religious Studies  
Arizona State University



## Course Description

This course is a study of the use of certain techniques to model intelligence. The techniques consist primarily in the application of logic within the context of an observation-thought-decision-action cycle. A prior course in symbolic logic (PHI 333 or equivalent) is helpful but not required.

At the completion of this course, students will be familiar with the general idea of computation and the idea that thinking is a form of computation, will

understand basic aspects of logic programming and its underlying theory, will be familiar with the application of logic programming and some of its variants within the context the agent model of intelligence, will understand basic aspects of the computer language Prolog ("PROgramming in LOGic"), will be familiar with basic aspects of natural language processing, will know some of the experiments in psychology that challenge the use of logic to model human intelligence, and will be familiar with some of the challenges to the logical approach to modeling intelligence generally.

CS1a  
CS16vi

This course satisfies CS (computer/statistics/quantitative applications) in the *University Undergraduate General Studies Requirement*. This course also satisfies a requirement for the *Symbolic Systems Certificate*. This certificate is modeled on the program at Stanford, the course of study taken by many influential figures in technology, such as Marissa Mayer (President and CEO of Yahoo!).

## Required Books

There are two required books for the course: Robert Kowalski's *Computational Logic and Human Thinking: How to be Artificially Intelligent* (Cambridge University Press, 2011) and Hector Levesque's *Thinking as Computation: A First Course* (MIT Press, 2012).

In addition, the authors have made public some of their teaching materials: [slides](#) for *Thinking as Computation*, slides for a [shorter](#) and a [longer](#) course based on *Computational Logic and Human*

*Thinking*, and [video](#) lectures (which were recorded at the 22nd International Joint Conference on Artificial Intelligence (IJCAI), Barcelona 2011) for *Computational Logic and Human Thinking*. These teaching materials are strictly supplementary to the books and my lecture notes.

## Grade for the Course

The final grade for the course is a function of your grade on 7 assignments. Each assignment (listed below) is worth 14 out of a total of 100 points. There are 2 free points. There is no extra credit. Attendance is not required, but passing the course is unlikely without regular class attendance. Keep your graded assignments. They are your only record of your grades. Incompletes are given only to accommodate serious illnesses and family emergencies, which must be adequately documented.

The final grade for the course uses plus-minus letter grades, A+ to E. This grade is a weighted averaged computed using ASU's numerical value for the letter grades (A+ = 4.3, A = 4.0, A- = 3.7, B+ = 3.3, B = 3.0, B- = 2.67, C+ = 2.33, C = 2.00, D = 1.00, E = 0).

Here is an example to illustrate how the final grade is computed:

Assignment #1, A-	$14\%(3.7) = .518$
Assignment #2, B+	$14\%(3.3) = .462$
Assignment #3, B	$14\%(3.0) = .42$
Assignment #4, A	$14\%(4.0) = .56$
Assignment #5, B	$14\%(3.0) = .42$
Assignment #6, B	$14\%(3.0) = .42$
Assignment #7, B	$14\%(3.0) = .42$
Free points, A+	$2\%(4.3) = .086$

In the example, the weighted average sums to 3.306. Relative to ASU's numerical values for letter grades, 3.306 is closest to B+ (= 3.3). So, in this example, the final grade for the course is B+.

## Lectures and Readings

Not all the following material is equally important. I highlight the most important points in my lectures and in my lecture notes. This (in addition to the reading in the books) is the material on which you will be graded in your assignments. Use the slides and video as supplementary. It is not necessary to understand every detail of the supplementary material. Some of this material is difficult and appropriate for a more advanced course. The lectures and lecture notes are more understandable than the books, and the books are more understandable than the slides and the video lectures.

## UNIT 1: ds la l bvi

### Thinking is Computation (The Hypothesis in the Course)

- *Thinking as Computation* 1, 2; slides 1-38.
- *Computational Logic and Human Thinking* "Introduction," 1; shorter 1-10; video 00:00-12:40; longer 1-33.

### Logic and Logic Programming (The Technical Background)

- *Thinking as Computation* 2; slides 39-55.
- *Computational Logic and Human Thinking* "Appendix A1," "Appendix A2," "Appendix A3," "Appendix A5"; shorter 11-15; video 11:04-18:59; longer 33-60, 75-95, 189-214.

### Assignment #1

## UNIT 2: ds la l bvi

### Prolog (A Computer Programming Language)

- *Thinking as Computation* 3, 4, "Appendix A," "Appendix B," "Appendix C," "Appendix D"; slides 56-97.

### The Psychology of Logic (The Wason Selection Task and The Suppression Task)

- *Computational Logic and Human Thinking* 2; longer 61-74.

### Assignment #2

## UNIT 3: ds la l bvi

### The Fox and the Crow (The Logic Programming/Agent Model)

- *Computational Logic and Human Thinking* 3, 8; longer 96-108, 157-188.

### Assignment #3

## UNIT 4: ds la l bvi

### Negation as Failure (The Suppression Task Revisited)

- *Computational Logic and Human Thinking* 5, "Appendix A4"; longer 109-130, 247-261.

### Assignment #4

## UNIT 5: *CS 1a 1bvi*

### Prohibitions and Prospective Logic Programming

- *Computational Logic and Human Thinking* 12.

### Abduction and Abductive Logic Programming

- *Thinking as Computation* 11; slides 314-322.
- *Computational Logic and Human Thinking* 10, "Appendix A6"; longer 215-228

### Assignment #5

## UNIT 6: *CS 1a 1bvi*

### Understanding Natural Language

- *Thinking as Computation* 8; slides 188-228.

### Assignment #6

## UNIT 7: *CS 1a 1bvi*

### The Wason Selection Task Revisited

- *Computational Logic and Human Thinking* 16.

### Assignment #7

## **Contact Information**

Thomas A. Blackson

Philosophy Faculty

School of Historical, Philosophical, and Religious Studies

Lattie F. Coor Hall, room 3356

PO Box 874302

Arizona State University

Tempe, AZ. 85287-4302

[blackson@asu.edu](mailto:blackson@asu.edu), [tomblackson.com](http://tomblackson.com), [www.public.asu/~blackson](http://www.public.asu/~blackson)

CS 1a, 1b vi

---

# Thinking as Computation

*A First Course*

Hector J. Levesque

The MIT Press  
Cambridge, Massachusetts  
London, England

# Contents

<b>Preface</b>	<b>xiii</b>
Background . . . . .	xiii
Overview of the book . . . . .	xvi
Guide for the course instructor . . . . .	xvii
Guide for the student . . . . .	xix
<b>Acknowledgments</b>	<b>xxi</b>
<b>1 Thinking and Computation</b>	<b>1</b>
1.1 Thinking . . . . .	2
1.1.1 What is thinking? . . . . .	3
1.2 Computation . . . . .	4
1.2.1 Symbols and symbolic structures . . . . .	4
1.2.2 What is computation? . . . . .	5
1.2.3 Some arithmetic procedures . . . . .	5
1.2.4 The lesson . . . . .	10
1.3 Thinking as computation . . . . .	11
1.3.1 Leibniz and his idea . . . . .	12
1.3.2 Propositions vs. sentences . . . . .	13
1.3.3 Using what is known: The web of belief . . . . .	16
Want to read more? . . . . .	20
Exercise . . . . .	21
<b>2 A Procedure for Thinking</b>	<b>23</b>
2.1 Atomic and conditional sentences . . . . .	23
2.2 Logical entailment . . . . .	24
2.3 Back-chaining . . . . .	27
2.3.1 Using variables . . . . .	27
2.3.2 Tracing the back-chaining . . . . .	28
2.4 Variables in queries . . . . .	31
2.4.1 One complication: Renaming variables . . . . .	31

2.4.2	Another complication: Backtracking . . . . .	33
2.4.3	A more complex query . . . . .	34
2.5	Why is back-chaining good? . . . . .	36
2.5.1	Getting stuck in a loop . . . . .	37
	Want to read more? . . . . .	38
	Exercises . . . . .	39
<b>3</b>	<b>The Prolog Language</b> . . . . .	<b>41</b>
3.1	Prolog programs . . . . .	42
3.2	Prolog queries . . . . .	45
3.2.1	Queries and their outcomes . . . . .	46
3.2.2	Conjunctive queries . . . . .	48
3.2.3	Negation in queries . . . . .	49
3.2.4	Tracing the back-chaining . . . . .	50
3.2.5	Instantiated and uninstantiated variables . . . . .	52
3.2.6	Equality in queries . . . . .	53
3.3	Prolog back-chaining . . . . .	55
3.3.1	Unification . . . . .	56
3.3.2	Renaming variables . . . . .	58
* 3.3.3	Back-chaining revisited . . . . .	58
	Want to read more? . . . . .	61
	Exercises . . . . .	61
<b>4</b>	<b>Writing Prolog Programs</b> . . . . .	<b>63</b>
4.1	The truth in Prolog . . . . .	63
4.1.1	The truth, and nothing but . . . . .	63
4.1.2	The whole truth . . . . .	64
4.2	A blocks world . . . . .	66
4.3	Recursion in Prolog . . . . .	68
* 4.4	Mathematical induction . . . . .	69
4.5	Nonterminating programs . . . . .	72
4.6	A more complex predicate . . . . .	75
* 4.6.1	Recursion and termination, reconsidered . . . . .	75
4.7	Efficiency in Prolog . . . . .	78
	Want to read more? . . . . .	81
	Exercises . . . . .	82

.....	33		
.....	34		
.....	36		
.....	37		
.....	38		
.....	39		
	41		
.....	42		
.....	45		
.....	46		
.....	48		
.....	49		
.....	50		
.....	52		
.....	53		
.....	55		
.....	56		
.....	58		
.....	58		
.....	61		
.....	61		
	63		
.....	63		
.....	63		
.....	64		
.....	66		
.....	68		
.....	69		
.....	72		
.....	75		
.....	75		
.....	78		
.....	81		
.....	82		
		<b>5 Case Study: Satisfying Constraints</b>	<b>85</b>
		5.1 Constraint satisfaction problems	86
		5.1.1 Generate-and-test	86
		5.1.2 Variables, domains, constraints	89
		5.1.3 Output in Prolog	89
		5.2 A first example: Sudoku	91
		5.2.1 The anonymous variable in Prolog	91
		5.2.2 Sudoku as constraint satisfaction	92
		5.2.3 Search spaces	94
		5.2.4 Guessed values and forced values	95
		5.3 A second example: Cryptarithmic	96
		5.3.1 Arithmetic in Prolog	96
		5.3.2 Cryptarithmic as constraint satisfaction	99
		5.3.3 Minimizing the guesswork: Two rules	101
		* 5.4 A third example: The eight queens	103
		5.5 A fourth example: Logic problems	107
		5.5.1 Hidden variables	108
		* 5.5.2 A more complex logic problem	109
		* 5.6 A fifth example: Scheduling	112
		Want to read more?	114
		Exercises	115
		<b>* 6 Case Study: Interpreting Visual Scenes</b>	<b>119</b>
		6.1 The thinking part of vision	119
		6.2 Aerial sketch maps	121
		6.2.1 Constraints on image regions	122
		6.3 Polyhedral objects	124
		6.3.1 Constraints on vertices and edges	126
		6.3.2 Impossible objects	130
		6.4 Object recognition	130
		* 6.4.1 Handling occlusion	132
		Want to read more?	135
		<b>7 Lists in Prolog</b>	<b>137</b>
		7.1 Lists	137
		7.1.1 Lists as Prolog terms	139
		7.1.2 Unification with lists	139



7.2	Writing programs that use lists	140
7.2.1	Some example list predicates	141
7.3	Using the member and append predicates	145
7.3.1	The blocks world revisited	149
	Want to read more?	150
	Exercises	150
<b>8</b>	<b>Case Study: Understanding Natural Language</b>	<b>153</b>
8.1	Analyzing the syntax of a language	154
8.1.1	Lexicon	155
8.1.2	Grammar	156
8.1.3	Parsing and ambiguity	157
8.2	Interpreting noun phrases	158
8.2.1	Writing a world model	159
8.2.2	Writing a lexicon	159
8.2.3	Writing a parser	163
8.2.4	Putting the pieces together	164
8.3	Interpreting sentences	170
8.3.1	Yes/no questions	170
8.3.2	Dynamic predicates in Prolog	172
8.3.3	Simple declarative sentences	173
8.4	Nonreferential noun phrases	174
	Want to read more?	175
	Exercises	176
<b>9</b>	<b>Case Study: Planning Courses of Action</b>	<b>179</b>
9.1	Planning problems	180
9.1.1	A first example: The three coins	180
9.1.2	A second example: The monkey and bananas	181
9.1.3	States and operators	182
9.2	Generating plans	183
9.2.1	A general planning program	183
9.2.2	Solving the three-coins problem	184
9.2.3	Atoms as terms in Prolog	185
9.2.4	Solving the monkey and bananas problem	186
9.2.5	Bounding plan length	187
9.2.6	A third example: The 15-puzzle	190

..... 140  
 ..... 141  
 ..... 145  
 ..... 149  
 ..... 150  
 ..... 150  
 ..... 153  
 ..... 154  
 ..... 155  
 ..... 156  
 ..... 157  
 ..... 158  
 ..... 159  
 ..... 159  
 ..... 163  
 ..... 164  
 ..... 170  
 ..... 170  
 ..... 172  
 ..... 173  
 ..... 174  
 ..... 175  
 ..... 176  
 ..... 179  
 ..... 180  
 ..... 180  
 as ..... 181  
 ..... 182  
 ..... 183  
 ..... 183  
 ..... 184  
 ..... 185  
 ..... 186  
 ..... 187  
 ..... 190

9.3 Scaling up: The search problem . . . . . 191  
 9.3.1 Knowledge-based planning . . . . . 192  
 \* 9.3.2 Best-first search . . . . . 195  
 9.4 Scaling up: The representation problem . . . . . 197  
 9.4.1 Situations and fluents . . . . . 198  
 9.4.2 Planning with situations and fluents . . . . . 202  
 9.4.3 Why is this representation useful? . . . . . 203  
 9.4.4 Other kinds of actions . . . . . 203  
 Want to read more? . . . . . 204  
 Exercises . . . . . 205

**10 Case Study: Playing Strategic Games . . . . . 209**  
 10.1 Games as problems . . . . . 210  
 10.1.1 How a game can be defined . . . . . 211  
 10.1.2 A first example: Race to 21 . . . . . 211  
 10.2 Finding the best move . . . . . 212  
 10.2.1 Game trees . . . . . 213  
 10.2.2 A general game player . . . . . 215  
 10.2.3 Playing Race to 21 . . . . . 217  
 10.2.4 A second example: Tic-tac-toe . . . . . 218  
 \* 10.2.5 Playing an entire game . . . . . 220  
 \* 10.2.6 A third example: Boxes . . . . . 222  
 10.3 Playing bigger games . . . . . 226  
 10.3.1 Numerical game trees and minimax . . . . . 227  
 10.3.2 Alpha and beta cutoffs . . . . . 231  
 10.3.3 The application to chess . . . . . 232  
 Want to read more? . . . . . 233  
 Exercises . . . . . 234

**\* 11 Case Study: Other Ways of Thinking . . . . . 239**  
 11.1 Back-chaining as subject matter . . . . . 241  
 11.1.1 A breadth-first thinking procedure . . . . . 244  
 \* 11.1.2 A forward-chaining thinking procedure . . . . . 245  
 11.1.3 Back-chaining with variables, negation, and equality . . . . . 247  
 11.2 Explanation . . . . . 249  
 11.2.1 Diagnosis . . . . . 249  
 11.2.2 A general explanation program . . . . . 251

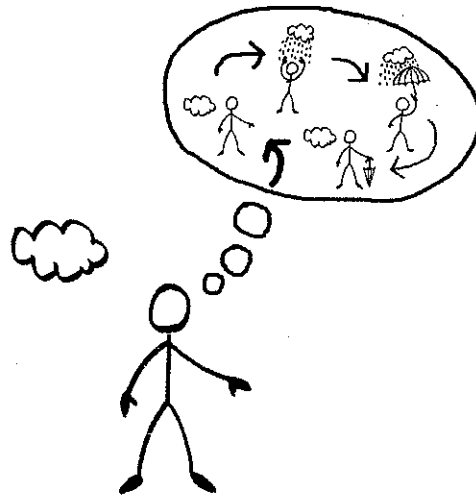
11.3 Learning . . . . .	253
11.3.1 Inducing general rules . . . . .	254
11.3.2 An example of induction . . . . .	254
* 11.3.3 Classification: Training and testing . . . . .	256
11.4 Propositional reasoning . . . . .	258
11.4.1 Conjunctive normal form . . . . .	259
11.4.2 Satisfiability . . . . .	260
11.4.3 Computing satisfiability . . . . .	261
11.4.4 Logical entailment reconsidered . . . . .	262
* 11.4.5 First-order reasoning . . . . .	265
Want to read more? . . . . .	267
<b>12 Can Computers Really Think?</b>	268
12.1 What computers can do . . . . .	270
12.2 The Turing Test . . . . .	271
12.3 The Chinese Room . . . . .	272
12.4 The Summation Room . . . . .	273
12.5 A final word . . . . .	274
Want to read more? . . . . .	275
<b>Appendix A Some Computer Basics</b>	275
A.1 Working with computer files . . . . .	276
A.2 Files available online . . . . .	279
<b>Appendix B Getting Started with SWI-Prolog</b>	279
B.1 Installing SWI-Prolog . . . . .	280
B.2 How to load SWI-Prolog programs . . . . .	283
<b>Appendix C Getting Your Prolog Programs to Work</b>	283
C.1 Getting program files to load . . . . .	285
C.2 Getting the right answers from queries . . . . .	286
C.2.1 Tracing the execution . . . . .	286
C.2.2 Interrupting the execution . . . . .	287
C.3 Saving a record of program execution . . . . .	287
<b>Appendix D Other Prolog Systems</b>	287
<b>References</b>	29
<b>Index of Technical Terms</b>	29

ds1a, 1b vi

# Computational Logic and Human Thinking

How to be Artificially Intelligent

ROBERT KOWALSKI  
*Imperial College London*



 **CAMBRIDGE**  
UNIVERSITY PRESS

# Contents

---

<i>Preface</i>	<i>page</i> ix
<i>Summary and plan of the book</i>	xii
<b>Introduction</b>	1
<b>1 Logic on the Underground</b>	9
<b>2 The psychology of logic</b>	24
<b>3 The fox and the crow</b>	40
<b>4 Search</b>	50
<b>5 Negation as failure</b>	60
<b>6 How to become a British Citizen</b>	77
<b>7 The louse and the Mars explorer</b>	92
<b>8 Maintenance goals as the driving force of life</b>	107
<b>9 The meaning of life</b>	125
<b>10 Abduction</b>	134
<b>11 The Prisoner's Dilemma</b>	144
<b>12 Motivations matter</b>	155
<b>13 The changing world</b>	166
<b>14 Logic and objects</b>	179
<b>15 Biconditionals</b>	188
<b>16 Computational Logic and the selection task</b>	198
<b>17 Meta-logic</b>	213
<b>Conclusions of the book</b>	227
<i>A1</i> The syntax of logical form	231
<i>A2</i> Truth	247
<i>A3</i> Forward and backward reasoning	257

<i>A4</i> Minimal models and negation	263
<i>A5</i> The resolution rule	269
<i>A6</i> The logic of abductive logic programming	280
<i>References</i>	296
<i>Index</i>	303