Computational Logic and Human Thinking

How to be Artificially Intelligent

The practical benefits of computational logic need not be limited to mathematics and computing. As this book shows, ordinary people in their everyday lives can profit from the recent advances that have been developed for artificial intelligence. The book draws upon related developments in various fields from philosophy to psychology and law. It pays special attention to the integration of logic with decision theory, and the use of logic to improve the clarity and coherence of communication in natural languages such as English.

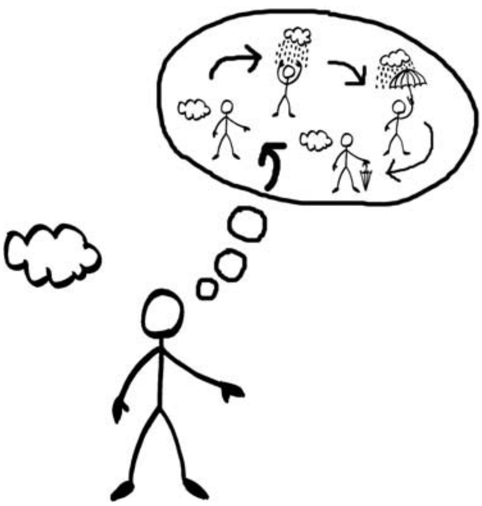
This book is essential reading for teachers and researchers who want to catch up with the latest developments in computational logic. It will also be useful in any undergraduate course that teaches practical thinking, problem solving or communication skills. Its informal presentation makes the book accessible to readers from any background, but optional, more formal, chapters are also included for those who are more technically oriented.

ROBERT KOWALSKI is Emeritus Professor and Research Investigator in the Department of Computing at Imperial College London. In 2011 he received the IJCAI Award for Research Excellence for his contributions to logic for knowledge representation and problem solving, including his pioneering work on automated theorem proving and logic programming.

Computational Logic and Human Thinking

How to be Artificially Intelligent

ROBERT KOWALSKI
Imperial College London



CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Tokyo, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521194822

First published 2011

Printed in the United Kingdom at the University Press, Cambridge

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data
Kowalski, Robert.
Computational logic and human thinking : how to be artificially
intelligent / Robert Kowalski.
p. cm.
ISBN 978-0-521-19482-2 (hardback)
1. Computational intelligence. 2. Logic, Symbolic and mathematical – Famous
problems. 3. Rhetoric – Mathematics. 4. Communication – Philosophy.
5. Reasoning. 6. Critical thinking. I. Title.
Q342.K69 2011
511.3–dc22
2011002461

ISBN 978-0-521-19482-2 Hardback
ISBN 978-0-521-12336-5 Paperback

To Bob, John and Mary

Contents

Preface

The mere possibility of Artificial Intelligence (AI) – of machines that can think and act as intelligently as humans – can generate strong emotions. While some enthusiasts are excited by the thought that one day machines may become more intelligent than people, many of its critics view such a prospect with horror.

Partly because these controversies attract so much attention, one of the most important accomplishments of AI has gone largely unnoticed: the fact that many of its advances can also be used directly by people, to improve their own human intelligence. Chief among these advances is Computational Logic.

Computational Logic builds upon traditional logic, which was originally developed to help people think more effectively. It employs the techniques of symbolic logic, which has been used to build the foundations of mathematics and computing. However, compared with traditional logic, Computational Logic is much more powerful; and compared with symbolic logic, it is much simpler and more practical.

Although the applications of Computational Logic in AI require the use of mathematical notation, its human applications do not. As a consequence, I have written the main part of this book informally, to reach as wide an audience as possible. Because human thinking is also the subject of study in many other fields, I have drawn upon related studies in Cognitive Psychology, Linguistics, Philosophy, Law, Management Science and English Composition.

In fact, the variant of Computational Logic presented in this book builds not only upon developments of logic in AI, but also upon many other complementary and competing knowledge representation and problem-solving paradigms. In particular, it incorporates procedural representations of knowledge from AI and Computing, production systems from AI and Cognitive Science, and decision analysis from Management Science, Cognitive Psychology and Philosophy.

Because Computational Logic has so many applications and so many relations with other fields, the ideal, ultimate use of this book would be as a companion text for an undergraduate degree in practical thinking. Such a degree course would combine the traditional virtues of a liberal arts education with the argumentation skills of analytic philosophy, the rigours of scientific method and the modern benefits of information technology. It would provide the student with the transferable thinking and communication skills needed not only for more specialised studies, but also for problems that do not fall into neatly classified areas.

As far as I know, nothing approaching such a degree course exists today; and as far as I can see, no such degree course is likely to exist in the near future. Logic as an academic discipline, as it exists today, is fragmented between Mathematics, Philosophy and Computing. Moreover, the practical applications of Informal Logic are mostly buried inside other academic disciplines, like Law, Management Science and English Composition. None of these disciplines could host such a degree course on its own, and few of them would welcome such an expansion of logic in their own field.

Perhaps one day, an educational institution will make room for a degree course focusing on how to think. In the meanwhile, this book can be used as a supplement to more conventional courses. For those who have already completed their formal education, it can provide a glimpse into a possible future world.

In writing this book, I have taken pains to avoid misrepresenting the subject by over-simplification. For this reason, I have included a number of additional, more advanced chapters, which fill in some of the otherwise missing technical detail. These chapters can be safely skipped by the casual reader. Taken on their own, they provide a self-contained introduction and reference to the formal underpinnings of the Computational Logic used in this book.

I have also been sensitive to the fact that, because I address issues of English writing style, I am inviting attention to the inadequacies of my own writing style. In defence, let me argue that without the help of Computational Logic, my writing would be a lot worse.

When I started my undergraduate studies at the University of Chicago years ago, my writing was so bad that I failed the placement examination and had to take an extra, non-credit, remedial course. I finished the year with As in all my other subjects, but with a D in English writing skills. It took me years to diagnose the problems with my writing and to learn how to improve it. In the course of doing so, I learned more about practical logic than I did in any of my formal logic courses. I like to believe that my writing is a lot better today than it was during my first year in Chicago. But more importantly, I hope that the lessons I learned will also be helpful to some of the readers of this book.

I am very grateful to Tom Blackson, François Bry, Tony Burton, Keith Clark, Jacinto Davila, Phan Minh Dung, Maarten van Emden, Steffen Hoelldobler, Luis Pereira, Yongyuth Permpoontanalarp, Fariba Sadri, Keith Stenning, Dania Kowalska-Taylor, Sten-Ake Tarnlund, Jeff Thompson, Francesca Toni and Mike Tunstall for their valuable comments on earlier drafts of the book. Thanks also to Simon Taylor for the drawing of the fox and the crow.

Summary and plan of the book

Because this book ranges over a wide variety of topics, it is useful to summarise the relationships between the different chapters in one place. However, instead of placing this summary at the end of the book, where all of its terms will have already been explained in detail, I have decided to present it here, in keeping with the general spirit of the book that it is better to work backwards from your destination, than to stumble forward, wondering where you are going.

Therefore, this summary may be read either before or after the main body of the book. But it can also be read in parallel, to get a better orientation of how the individual chapters are related.

Introduction

In Artificial Intelligence, an *agent* is any entity, embedded in a real or artificial world, that can observe the changing world and perform actions on the world to maintain itself in a harmonious relationship with the world. Computational Logic, as used in Artificial Intelligence, is the agent's language of thought. Sentences expressed in this language represent the agent's beliefs about the world as it is and its goals for the way it would like it to be. The agent uses its goals and beliefs to control its behaviour.

The agent uses the inference rules of Computational Logic, applying them to its thoughts in logical form, to reason about the world and to derive actions to change the world for its own benefit. These inference rules include both forward reasoning to derive consequences of its observations, and backward reasoning to reduce its goals to subgoals and actions. The agent can also use forward reasoning to deduce consequences of candidate actions, to help it choose between alternative candidates.

Although the main purpose of Computational Logic is to represent an agent's private thoughts and to control its behaviour, the agent can also use Computational Logic to guide its public communications with other agents. By expressing its communications in a more logical form, a speaker or writer can make it easier for the listener or reader to translate those communications into thoughts of her own.

Chapter 1: Logic on the Underground

The London Underground Emergency Notice illustrates the way in which the meanings of English communications can be understood as thoughts in logical form. In Computational Logic, these thoughts have both a logical and computational character. Their logical character is apparent in their explicit use of logical connectives, like *any*, *if*, *and* and *not*; and their computational character is manifest in their use as procedures for reducing goals to subgoals. Because of this dual logical and computational character, sentences expressed in this form are also called *logic programs*.

The Emergency Notice also illustrates how the coherent use of English communications can be understood in terms of logical connections between the meanings of those communications and other thoughts in an agent's web of goals and beliefs. Once the agent has made the connections, the agent can activate them by forward or backward reasoning, when the need arises. Connections that are activated frequently can be collapsed into derived goals or beliefs, which can be used more directly and more efficiently in the future.

Chapter 2: The psychology of logic

The most influential and widely cited argument against logic comes from psychological experiments about reasoning with natural-language sentences in conditional form. The most popular interpretation of these experiments is that people do not have a natural general-purpose ability to reason logically, but have developed instead, through the mechanisms of Darwinian evolution, specialised algorithms for solving typical problems that arise in their environment.

In this chapter I discuss some of the issues involved in solving these reasoning tasks, and argue that one of the main problems with the experiments is that they fail to appreciate that the natural-language form of a conditional is only an approximation to the logical form of its intended meaning. Another problem is that the interpretation of these experiments is based upon an inadequate understanding of the relationship between knowledge and reasoning. In Computational Logic applied to human thinking, this relationship can be expressed rather loosely as an equation: *thinking* = *specialised knowledge* + *general-purpose reasoning*.

Chapter 3: The fox and the crow

Aesop's fable of the fox and the crow illustrates the backward reasoning of a clever fox, to generate a plan to achieve the goal of having the cheese of a not so clever crow. It contrasts the fox's *proactive*, backward reasoning with the crow's *reactive*, forward reasoning, to respond to the fox's praise by breaking out in song, thereby dropping the cheese to the ground, where the fox can pick it up. Both the fox and the crow reason in accordance with the inference rules of Computational Logic, but the fox has a better knowledge of the world, and has more powerful ways of using that knowledge for her own benefit.

If the crow knew as much as the fox and were able to reason *preactively*, thinking before he acts, then he could reason forward from the hypothetical performance of his candidate actions, predict their likely consequences and choose an alternative action, like flying away or swallowing the cheese, that achieves a better expected resulting state of affairs.

Chapter 4: Search

In Computational Logic, a *proof procedure* consists of a collection of inference rules and a search strategy. The inference rules determine both the structure of proofs and the *search space* of all possible proofs relevant to the solution of a goal. The *search strategy* determines the manner in which the search space is explored in the search for a solution.

Many different search strategies are possible, including both parallel strategies, which explore different parts of the search space at the same time, and best-first strategies, which aim to find the best solution possible in the shortest amount of time.

Chapter 5: Negation as failure

In the semantics of Computational Logic, the world is a positive place, characterised by the positive atomic sentences that are true at the time. Because the ultimate purpose of an agent's goals and beliefs is to manage its interactions with the world, the syntactic form of the agent's thoughts also has a corresponding positive bias. In many cases, syntatically negative thoughts arise from the *failure* to observe or derive positive information.

Negation as failure is a natural way to reason by *default* with incomplete information, deriving conclusions under the assumption that the agent knows it all, but then gracefully withdrawing those conclusions if new information shows that they do not hold. It also facilitates higher-level ways of organising goals and beliefs into hierarchies of rules and exceptions, in which the rules represent only the most important conditions, and the exceptions add extra conditions when they are needed.

Chapter 6: How to become a British Citizen

The British Nationality Act is a body of English sentences, which states precisely the conditions under which a person may acquire, renounce or be deprived of British Citizenship. The Act is designed to be both unambiguous, so there is little doubt about its intended meaning, and flexible, so that it can be applied to changing circumstances. Its English style resembles the conditional form of sentences in Computational Logic.

In addition to its use of conditional form, the British Nationality Act illustrates many other important features of Computational Logic, including the representation of rules and exceptions, and meta-level reasoning about what it takes for a person, like you or me, to satisfy the Secretary of State that the person fulfils the requirements for naturalisation as a British Citizen.

In contrast with the British Nationality Act, the University of Michigan Lease Termination Clause shows how an ambiguous, virtually unintelligible English text can be made understandable by reformulating it in Computational Logic style.

Chapter 7: The louse and the Mars explorer

Arguably, the most influential computational model of human thinking in Cognitive Psychology is the production system model, as illustrated in this chapter by the wood louse and the Mars explorer robot. Production systems combine a *working memory* of atomic facts with *condition–action rules* of the form *if conditions then actions*. The working memory is like a model of the current state of the world, and the rules are like an agent's goals and beliefs.

The condition–action rules are embedded in an *observation–thought–decision–action cycle* and are executed by matching the conditions of rules with facts in the working memory and generating the actions of rules as candidate actions. This manner of execution is called *forward chaining*, which is similar to forward reasoning. If more than one candidate action is generated in this way, then a process, called *conflict resolution*, is used to decide between the candidates. The chosen action is then executed, changing the state of the working memory, simulating the way an agent's actions change the state of the world.

From a logical point of view, there are three kinds of condition–action rules: *reactive rules*, which are like instinctive stimulus–response associations; *goal-reduction rules*, which reduce goals to subgoals by forward chaining; and *forward reasoning rules*, which perform genuine logical forward reasoning.

Chapter 8: Maintenance goals as the driving force of life

The agent model presented in this book combines the functionalities of logic and production systems in a logical framework. The framework takes from production systems the *observation–thought–decision–action cycle*, but replaces condition–action rules by goals and beliefs in the logical form of conditionals. It replaces reactive rules by *maintenance goals* used to reason forwards, goal-reduction rules by *beliefs* used to reason backwards, and forward reasoning rules by *beliefs* used to reason forwards.

In the logical agent model, the agent *cycle* responds to observations of the environment by reasoning forwards with beliefs, until it derives a conclusion that matches one of the conditions of a maintenance goal. It reasons backwards, to check the other conditions of the maintenance goal. If all the conditions of the maintenance goal are shown to hold in this way, it reasons forwards one step, deriving the conclusion of the maintenance goal as an *achievement goal*. It then starts to reason backwards using its beliefs to reduce the achievement goal to a plan of candidate actions. It decides between different candidate actions, and starts to execute a plan. If necessary, it interrupts the execution of the plan, to process other observations, interleaving the plan with other plans.

Chapter 9: The meaning of life

The logical framework of the preceeding chapter views an agent's life as controlled by the changes that take place in the world, by its own goals and beliefs, and by the choices the agent makes between different ways of achieving its goals. The combination of its beliefs and its highest-level goals generates a hierarchy of goals and subgoals. However, for the sake of efficiency, this hierarchy may be collapsed into a collection of more direct stimulus–response associations, whose original goals are no longer apparent, but are implicit and emergent.

In AI and computing more generally, it is common for an intelligent designer to implement an artificial agent that does not contain an explicit representation of its higher-level goals. The designer is aware of the agent's goals, but the agent itself is not. As far as the agent is concerned, its life may seem to be entirely meaningless.

In this chapter, we contrast the seemingly meaningless life of an imaginary, artificial wood louse, with the more meaningful life of an intelligent agent, in which stimulus–response associations and awareness of higher-level goals are combined.

Chapter 10: Abduction

One of the main functions of an agent's beliefs is to represent causal relationships between its experiences. The agent uses these causal representations both *proactively* to generate plans to achieve its goals, and *preactively* to derive consequences of candidate actions to help it choose between alternative candidate actions. However, the agent can also use the same causal beliefs *abductively* to generate hypotheses to explain its observations, and to derive consequences of candidate hypotheses to help it choose between alternative hypotheses. This process of generating and choosing hypotheses to explain observations is called *abduction*.

Like default reasoning with negation as failure, abduction is *defeasible* in the sense that new information can cause a previously derived conclusion to be withdrawn.

Chapter 11: The Prisoner's Dilemma

The problem of deciding between alternative abductive explanations of an observation is similar to the problem of deciding between alternative actions, which is exemplified by the Prisoner's Dilemma. In this chapter, we see how an agent can use a combination of Computational Logic and decision theory to decide between alternatives. According to decision theory, the agent should choose an alternative that has the best expected outcome. The expected outcome of an action is determined by appropriately combining judgements of the utility (or desirability) of the action's consequences with judgements of the probability (or likelihood) that the consequence will actually happen.

Decision theory is a normative theory, which requires detailed knowledge of utilities and probabilities, but neglects the motivations of an agent's actions. In practice, agents more typically employ heuristic goals and beliefs (or rules of thumb), which approximate the decision-theoretic norms. But heuristics often go astray. When it is important to make smarter choices, it is better to use the more encompassing framework of the agent cycle, to analyse the motivations of actions and to ensure that a full range of alternatives is explored.

Chapter 12: Motivations matter

Decision theory leads to consequentialist theories of morality, which judge the moral status of actions simply in terms of their consequences. But in psychological studies and the law, people judge actions both in terms of their consequences and in terms of their motivations. We show how Computational Logic can model such moral judgements by using constraints to prevent actions that are deemed to be morally or legally unacceptable.

Chapter 13: The changing world

An agent's life is a continuous struggle to maintain a harmonious relationship with the ever-changing world. The agent assimilates its observations of the changing state of the world, and it performs actions to change the world in return.

The world has a life of its own, existing only in the present, destroying its past and hiding its future. To help it survive and prosper in such a changing environment, an intelligent agent uses beliefs about cause and effect, represented in its language of thought. In this chapter we investigate in greater detail the logical representation of such causal beliefs and the semantic relationship between this logical representation and the changing world.

Chapter 14: Logic and objects

Whereas in Cognitive Psychology production systems are the main competitor of logic, in Computing the main competitor is Object-Orientation. In the object-oriented way of looking at the world, the world consists of objects, which interact by sending and receiving messages. Objects respond to messages by using encapsulated methods, invisible to other objects, and inherited from general classes of objects.

Computational Logic is compatible with Object-Orientation, if objects are viewed as agents, methods are viewed as goals and beliefs, and messages are viewed as one agent supplying information or requesting help from another. Viewed in this way, the main contribution of Object-Orientation is twofold: it highlights the value both of structuring knowledge (goals and beliefs) in relatively self-contained modules, and of organising that knowledge in abstract hierarchies.

Chapter 15: Biconditionals

In this chapter we explore the view that conditional beliefs are biconditionals in disguise. For example, given *only* the two alternative conditions that can cause an object to look red:

an object looks red if the object is red.
an object looks red if it illuminated by a red light.

the two conditionals can be understood as standing for the biconditional:

an object looks red ***if and only if***
the object is red **or** *the object is illuminated by a red light.*

Both negation as failure and abduction can be understood as reasoning with such biconditionals as equivalences, replacing atomic formulas that match the conclusion by the disjunction of conditions (connected by ***or***) that imply the conclusion.

Chapter 16: Computational Logic and the selection task

In this chapter we return to the problem of explaining some of the results of psychological experiments about reasoning with conditionals. We investigate the different ways that Computational Logic explains these results, depending on whether a conditional is interpreted as a goal or as a belief. If it is interpreted as a belief, then it is often natural to interpret the conditional as specifying the *only* conditions under which the conclusion holds. This explains one of the two main mistakes that people make when reasoning with conditionals, when judged by the standards of classical logic.

The other main mistake is that people often fail to reason correctly with negation. This mistake is explainable in part by the fact that an agent's observations are normally represented by positive atomic sentences, and that negative conclusions have to be derived from positive observations. In many cases this derivation is easier with conditional goals than with conditional beliefs.

Chapter 17: Meta-logic

In this chapter we explore how meta-logic can be used to simulate the reasoning of other agents, and to solve problems that cannot be solved in the object language alone. We illustrate this with a variant of the wise man puzzle, and with Gödel's theorem that there are true but unprovable sentences in arithmetic.

Conclusions of the book

This concluding chapter takes a step back from the details, and takes a broader look at the main aim of the book, which is to show how Computational Logic can reconcile conflicting paradigms for explaining and guiding human behaviour. It also suggests how Computational Logic may help to reconcile conflicts in other areas.

Chapter A1: The syntax of logical form

This additional, more formal chapter gives a more precise formulation of Computational Logic as a logic of sentences having the conditional form *if conditions then conclusion* or equivalently having the form *conclusion if conditions*. In its simplest form, the *conclusion* of a conditional is an *atomic expression*, consisting of a *predicate* and a number of *arguments*. The *conditions* are a conjunction (connected by *and*) of atomic expressions or the *negations* of atomic expressions.

In this chapter, I compare the conditional form of logic with standard classical logic. I argue that classical logic is to conditional logic, as natural language is to the language of thought. In both cases, there are two kinds of reasoning, performed in two stages. The first stage translates sentences that are unstructured and possibly difficult to understand into simpler sentences that are better structured. The second stage derives consequences of the resulting simpler sentences. The logic of conditional forms is the logic of such simpler and better structured sentences.

Chapter A2: Truth

Conditionals in Computational Logic represent an agent's goals and beliefs in its private language of thought. They also represent the meanings of its public communications with other agents, and for this reason they can be said to represent the *semantics* of natural-language sentences. However, sentences in logical form also have a *semantics* in terms of their relationship with states of the world.

This additional chapter makes a start on the discussion of this semantics, and of the relationship between truth in all models and truth in minimal models. It argues from the example of arithmetic that truth in minimal models is more fundamental than truth in all models.

Chapter A3: Forward and backward reasoning

This chapter defines the forward and backward rules of inference more precisely, and shows how they can be understood in semantic terms, as showing how the truth of one set of sentences implies the truth of another. This semantic point of view applies both to the use of these inference rules to determine truth in all models and to their use to generate and determine truth in minimal models.

Chapter A4: Minimal models and negation

This chapter shows how the semantics of negation as failure can be understood in terms of the minimal model semantics of Chapter A2.

Chapter A5: The resolution rule of inference

In this chapter we see that forward and backward reasoning are both special cases of the resolution rule of inference, and that resolution is the underlying mechanism for reasoning in connection graphs.

Resolution was originally presented as a machine-oriented rule of inference, whereas forward and backward reasoning are human-oriented ways of understanding human thinking. This combination of human and machine orientation is reflected in the fact that the human mind can be regarded as a computing machine whose software is a conditional form of logic and whose hardware is a connectionist form of resolution.

Chapter A6: The logic of abductive logic programming

This chapter provides most of the technical support for the combination of forward reasoning, backward reasoning and negation as failure, which are the basic inference rules of the Computational Logic used in this book.

The proof procedure presented in this chapter can be understood in semantic terms, as generating a minimal model in which an agent's goals and beliefs are all true. However, it can also be understood in argumentation terms, as generating an argument in favour of a claim, both by providing support for the claim and by defeating all attacking arguments with counter-arguments.