KNOWLEDGE REPRESENTATION, REASONING AND DECLARATIVE PROBLEM SOLVING

KNOWLEDGE REPRESENTATION, REASONING AND DECLARATIVE PROBLEM SOLVING

CHITTA BARAL
Arizona State University

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2003

Printed in the United Kingdom at the University Press, Cambridge

Typeface Times 11/14 pt *System* LaTeX 2$_\varepsilon$ [TB]

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

Library of Congress Cataloguing in Publication data
Baral, Chitta.
Knowledge representation, reasoning, and declarative problem solving / Chitta Baral.
p. cm.
Includes bibliographical references and index.
ISBN 0 521 81802 8
1. Expert systems (Computer science) 2. Artificial intelligence. 3. Knowledge representation (Information theory). I. Title.
QA76.76.E95 B265 2002
006.3′3–dc21 2002025622

ISBN 0 521 81802 8 hardback

Contents

Preface

Representing knowledge and reasoning with it are important components of an intelligent system, and are two important facets of Artificial Intelligence. Another important expectation from intelligent systems is their ability to accept high level requests – as opposed to detailed step-by-step instructions, and their knowledge and reasoning ability are used to figure out the detailed steps that need to be taken. To have this ability intelligent systems must have a declarative interface whose input language must be based on logic.

Thus the author considers the all-round development of a suitable declarative knowledge representation language to be a fundamental component of knowledge based intelligence, perhaps similar to the role of the language of calculus to mathematics, and physics. Taking the calculus analogy further, it is important that a large support structure is developed around the language, similar to the integration and derivation formulas and the various theorems around calculus.

Although several languages have been proposed for knowledge representation, the language of AnsProlog* – logic programming with the answer set semantics, stands out in terms of the size and variety of the support structure developed around it. The support structure includes both implementations and use of the implementations in developing applications, and theoretical results for both analyzing and step-by-step building of theories (or programs) in this language. The support structure and the desirable properties of the language are also a testimony to the appropriateness of the language for knowledge representation, reasoning, and declarative problem solving.

This book is about AnsProlog* and compiles the various results obtained over the years about AnsProlog*. This book is expected to be useful to researchers in logic programming, declarative programming, artificial intelligence, knowledge representation, and autonomous agents; to knowledge engineers who would like to create and use large knowledge bases; to software practitioners who would like to use declarative programming for fast prototyping, and for developing critical

programs that must be correct with respect to a formal specification; to programmers of autonomous agents who would like to build intelligent components such as planners, schedulers, and diagnosis and repair systems; and to students and teachers using it as a text book in undergraduate and graduate classes.

The distinguishing features of this book are: (i) It uses answer set semantics of logic programs. (ii) A big part of this book is about declarative programming and knowledge representation methodology. It presents several small and big example modules, and presents the theory that describes when modules can be combined, when a module is consistent, how to incorporate an observation, etc. (iii) Because it uses answer set semantics which allows multiple 'models' of a theory, it is able to go beyond reasoning to declarative problem solving. Thus it includes encoding of applications such as planning, diagnosis, explanation generation, scheduling, combinatorial auctions, abductive reasoning, etc. Most of these applications are related to encoding problems that are **NP**-complete or beyond. The book also explores the well-founded semantics. Since the well-founded semantics is sound with respect to answer set semantics and is easier to compute, in this book it is treated as an approximation to answer set semantics. (iv) The book discusses complexity and expressiveness issues and identifies subsets belonging to different complexity and expressiveness classes. (v) It presents algorithms to compute answer sets. Some of the algorithms it discusses use heuristics and other intelligent search ideas. (vi) Most of the programs discussed in the book can be run. It uses the smodels and the dlv interpreter for this and is supplemented by a web site containing a large subset of the example programs as smodels or dlv code. We now give a brief description of the various chapters of the book.

0.1 Brief description of the chapters

- **Chapter 1: Declarative programming in AnsProlog*: introduction and preliminaries**
  In Chapter 1 we motivate the importance of declarative languages and argue that intelligent entities must be able to comprehend and process descriptions of '*what*', rather than being told '*how*' all the time. We then make the case for AnsProlog* (programming in logic with answer set semantics) and compare it with other nonmonotonic languages, and with the PROLOG programming language. We then present the syntax and semantics of various sub-classes of AnsProlog*, and consider two views of AnsProlog* programs: stand alone programs, and functions. There are more than 30 examples illustrating the various definitions and results.
- **Chapter 2: Simple modules for declarative programming with answer sets**
  In this chapter we present several small AnsProlog* programs or modules corresponding to several problem solving or knowledge representation modules. This chapter is like a tool box of programs that can be combined for larger applications. In a sense it gives a quick glimpse of the book, and can be thought of as introducing the usefulness and applicability of AnsProlog* through examples.

- **Chapter 3: Principles and properties of declarative programming with answer sets**
  In this chapter we present several fundamental results that are useful in *analyzing* and *step-by-step building* of AnsProlog* programs, viewed both as stand alone programs and as functions. To analyze AnsProlog* programs we define and describe several properties such as categoricity (presence of unique answer sets), coherence (presence of at least one answer set), computability (answer set computation being recursive), filter-abducibility (abductive assimilation of observations using filtering), language independence (independence between answer sets of a program and the language), language tolerance (preservation of the meaning of a program with respect to the original language when the language is enlarged), strong equivalence, compilability to first-order theory, amenability to removal of **or**, and restricted monotonicity (exhibition of monotonicity with respect to a select set of literals).

  We also define several sub-classes of AnsProlog* programs such as stratified, locally stratified, acyclic, tight, signed, head cycle free and several conditions on AnsProlog* rules such as well-moded, and state results about which AnsProlog* programs have what properties. We present several results that relate answer sets of an AnsProlog* program to its rules. We develop the notion of splitting and show how the notions of stratification, local stratification, and splitting can be used in step-by-step computation of answer sets.

  For *step-by-step building* of AnsProlog* programs we develop the notion of conservative extension – where a program preserves its original meaning after additional rules are added to it, and present conditions for programs that exhibit this property. We present several operators such as incremental extension, interpolation, domain completion, input opening, and input extension, and show how they can be used for systematically building larger programs from smaller modules.
- **Chapter 4: Declarative problem solving and reasoning in AnsProlog***
  In this chapter we formulate several knowledge representation and problem solving domains using AnsProlog*. Our focus in this chapter is on program development. We start with three well-known problems from the literature of constraint satisfaction, and automated reasoning: placing queens on a chess board, determining who owns the zebra, and finding tile covering in a mutilated chess board. We present several encodings of these problems using AnsProlog* and analyze them. We then discuss a general methodology for representing constraint satisfaction problems (CSPs) and show how to extend it to dynamic CSPs. We then present encodings of several combinatorial graph problems such as k-colorability, Hamiltonian circuit, and k-clique. After discussing these problem solving examples, we present a general methodology of reasoning with prioritized defaults, and show how reasoning with inheritance hierarchies is a special case of this.
- **Chapter 5: Reasoning about actions and planning in AnsProlog***
  In this chapter we consider reasoning about actions in a dynamic world and its application to plan verification, simple planning, planning with various kinds of domain constraints, observation assimilation and explanation, and diagnosis. We make a detailed and systematic formulation – in AnsProlog* – of the above issues starting from the simplest reasoning about action scenarios and gradually increasing its expressiveness by adding features such

as causal constraints, and parallel execution of actions. We also prove properties of our AnsProlog* formulations using the results in Chapter 3.

Our motivation behind the choice of a detailed formulation of this domain is two fold. (i) Reasoning about actions captures both major issues of this book: knowledge representation and declarative problem solving. To reason about actions we need to formulate the frame problem whose intuitive meaning is that objects in the worlds do not normally change their properties. Formalizing this has been one of the benchmark problems of knowledge representation and reasoning formalisms. We show how AnsProlog* is up to this task. Reasoning about actions also forms the ground work for planning with actions, an important problem solving task. We present AnsProlog* encodings of planning such that each of the answer sets encodes a plan. (ii) Our second motivation is in regard to the demonstration of the usefulness of the results in Chapter 3. We analyze and prove properties of our AnsProlog* formulations of reasoning about actions and planning by using the various results in Chapter 3, and thus illustrate their usefulness. For this we also start with simple reasoning about action scenarios and then in latter sections we consider more expressive scenarios.

- **Chapter 6: Complexity, expressiveness and other properties of AnsProlog* programs**
  In this chapter we consider some broader properties that help answer questions such as: (a) how difficult is it to compute answer sets of various sub-classes of AnsProlog*? (b) how expressive are the various sub-classes of AnsProlog*? (c) how modular is AnsProlog*? and (d) what is the relationship between AnsProlog* and other non-monotonic formalisms?

  The answers to these questions are important in many ways. For example, if we know the complexity of a problem that we want to solve then the answer to (a) will tell us which particular subset of AnsProlog* will be most efficient, and the answer to (b) will tell us the most restricted subset that we can use to represent that problem. To make this chapter self complete we start with the basic notions of complexity and expressiveness, and present definitions of the polynomial, arithmetic and analytical hierarchy, and their normal forms. We later use them to show the complexity and expressiveness of AnsProlog* subclasses.
- **Chapter 7: Answer set computing algorithms**
  In this chapter we present several answer set computing algorithms and compare them. The particular algorithms we present are the wfs-bb algorithm that uses branch and bound after computing the well-founded semantics, the assume-and-reduce algorithm of SLG, the smodels algorithm, and the dlv algorithm.
- **Chapter 8: Query answering and answer set computing systems**
  In this chapter we explain how to program using the Smodels and dlv systems, discuss the extensions that these systems have beyond AnsProlog*, and present several programs in their syntax. We then describe when a PROLOG interpreter can be used in answering queries to AnsProlog* programs and under what conditions the PROLOG interpreter is sound and complete with respect to AnsProlog*. We present several applications developed using the Smodels and dlv systems. This includes, combinatorial auctions, planning with durative actions and resources, scheduling, and specification and verification of active databases.

- **Chapter 9: Further extensions of and alternatives to AnsProlog***
  In this chapter we discuss further extensions to AnsProlog*, such as allowing **not** in the head of rules, allowing nested expressions, allowing epistemic operators, doing abductive reasoning, allowing set constructs, and allowing specification of priorities between rules. We discuss the universal query problem and discuss an extension where domain closure can be selectively specified. We also discuss some of the alternative characterizations of programs in AnsProlog* syntax.
- **Appendices**
  There are two small appendices in the book, one about ordinals, lattices, and fixpoints, and another on Turing machines.
- **Web site**
  The web site of the book has the Smodels and dlv code of several programs discussed throughout the book, a list of pointers to resources such as home pages of active scientists and researchers in the field, implemented systems, and applications and programs written with respect to those systems. See http://www.baral.us/bookone

0.2 Using it as a textbook

The book is a systematic compilation of the available support structure around AnsProlog*. It can be used as a text book with some selection and reordering of the material by the instructor. For example, for an undergraduate (junior-senior) course it is recommended that most of Chapter 1, Chapter 2, a very small part of Chapter 3, Chapter 4, parts of Chapter 5, parts of Chapter 7, and Chapter 8 are covered. In this the Smodels and dlv system in Chapter 8 should be introduced concurrently with Chapter 1 so that as the students learn the definition, they can program it. Many of the notations in Chapter 1 should first be glanced over and students should come back to it when necessary.

For a follow-up graduate course it is recommended that the remaining material (Chapter 3, most of Chapter 5, Chapter 6, parts of Chapter 7, and Chapter 9) are covered together with a quick overview of Chapters 1, 2, 4, and 8. For a stand alone graduate course that does not have the undergraduate course as the pre-requisite it is recommended that Chapters 1–8 are covered, leaving out a few sections in some of the chapters.

0.3 Appreciation and thanks

This book came about after almost a decade of interacting with and learning from my colleague and teacher Michael Gelfond at the University of Texas at El Paso. Many ideas and perspectives behind this book that unify the different sections and chapters are due to this interaction and learning. Interactions with Vladimir Lifschitz and his feedback also had a significant influence in the various aspects

of this book. Thomas Eiter helped the author to understand several aspects of the complexity and expressiveness results and gave detailed feedback on Chapter 6. Gerald Pfeifer gave detailed feedback on the dlv algorithm and the dlv system covered in Chapters 7 and 8. Mauricio Osorio gave feedback on Chapters 1–5. Jack Minker (my doctoral advisor) gave feedback on several aspects of the book. Jürgen Dix provided help in the writing of Section 6.6.9, Mauricio Osorio helped in the writing of Section 3.9.7, Kewen Wang and Torsten Schaub helped in the writing of Section 9.7, Phan Minh Dung helped in the writing of Section 9.8.5, Jia-Huai You helped in the writing of Section 9.8.6, and Mutsumi Nakamura helped in the writing of Section 8.3.5.

The author would like to acknowledge the help of his students (Nam Tran, Tran Son, Le-Chi Tuan, Viet Hung Nguyen, Cenk Uyan, Saadat Anwar and Guray Alsac), the Smodels group, the dlv group, the Texas action group (TAG), and the encouragement of his colleagues Jorge Lobo, Fang-Zhen Lin, Alessandro Provetti, Mirek Truszczyǹski, Jose Alferes, Vladik Kreinovich, Luís Moniz Pereira, Sheila McIlraith, Yan Zhang, Ramon Otero, Pedro Cabalar, Dan Cooke, Ken Ford, Rao Kambhampati, Huan Liu and Hudson Turner in writing this book. He would also like to acknowledge support from his start-up funds at the Arizona State University, support of his past Chair Stephen Yau, and NSF support through grants IRI-9501577 and 0070463 and NASA support through grant NCC2-1232.

The author appreciates the support of David Tranah and Cambridge University Press through the whole process of getting this book out. In particular, the copy editing significantly improved the text.

Finally, the author would like to thank his parents, extended family (siblings, in-laws, Jay, and Sabrina) and his spouse Mutsumi for their support and patience, and the loving and supporting environment they created when the book was being written.