Knowledge representation, reasoning and declarative problem solving with Answer sets[1]

Chitta Baral
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287
chitta@asu.edu

July 31, 2001

[1]NOTE: THIS IS A DRAFT. I PLAN TO HAVE TWO MORE ITERATIONS FROM BEGINNING TO END. AMONG THINGS THAT REMAIN TO BE DONE ARE: HAVING A PROPER NAMING CONVENTION FOR THE PROGRAMS IN THE EXAMPLES, CLEANING UP ENGLISH ERRORS, FILLING-IN MISSING SECTIONS, ETC.

Preface

This book is about the language of logic programming with answer set semantics and its application to knowledge representation, reasoning and declarative problem solving. This book will 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 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 bulk of the book focuses on the research in logic programming since 1988, when the *stable model* semantics was proposed by Gelfond and Lifschitz. The main differences between this book and earlier books[1] on logic programming (besides the fact that it includes results since those earlier books were written) are: (i) It uses answer set semantics and covers all programs. In contrast the book by Lloyd covers only stratified programs, and the book by Lobo et al. and Alferes et al. use well-founded semantics or its variant with respect to non-stratified 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. To the best of our knowledge no book on logic programming discusses this. (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 well-founded semantics used in the other books is less expressive and is mostly suitable for reasoning tasks. The book does devote some attention to 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 the complexity and expressibility issues and identifies subsets belonging to different complexity and expressibility classes. (v) It presents algorithms to compute answer sets. Some of the algorithms it discusses use heuristics and other intelligent search ideas. (vi) Unlike the books by Lloyd and Lobo et al. 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.

This book can be used for an undergraduate (junior-senior level) one semester course and a one semester graduate course. For the undergraduate course it is recommended to cover 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. For a follow-up graduate course it is recommended to cover the remaining material (Chapter 3, most of Chapter 5, Chapter 6, parts of Chapter 7, and Chapter 9) together with a quick overview of Chapters 1, 2, 4 and 8. For a stand alone graduate course that does not have

[1] The earlier books that we are referring to are: the books by Lloyd published in 1982 (first edition) and 1987 (second edition) titled 'Foundations of logic programming', the monograph by Lobo, Minker and Rajasekar published in 1992 titled 'Foundations of disjunctive logic programming', and the monograph by Alferes and Pereira published in 1996 titled 'Reasoning with logic programming'.

the undergraduate course as the pre-requisite it is recommended to cover Chapter 1-8, leaving out a few sections in some of the chapters.

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. Thomas Eiter helped the author in understanding several aspects of the complexity and expressibility results. Interactions with Vladimir Lifschitz had a significant influence in the various aspects of this book. The author would also like to acknowledge the help of his students (Nam Tran, Tran Son, Le-Chi Tuan, Viet Hung Nguyen, Cenk Uyan, and Guray Alsac), the smodels group (particularly, Niemela and Syrjanen), dlv group (particularly Eiter, Gottlob, Leone, and Pfiefer), and his colleagues Lobo, Provetti and Galindo in writing this book. Finally, he would like to acknowledge support from his start-up funds at the Arizona State University, NSF support through grants IRI-9501577 and 0070463 and NASA support through grant NCC2-1232.

About the book

Representing knowledge and reasoning with it are important components of an intelligent system, and are two important facets of Artificial Intelligence. In this the role of formal logic, in particular the non-monotonic logics is well established. Another important expectation from intelligent systems is their ability to accept high level requests – as opposed to detailed step-by-step instructions, and use their knowledge and reasoning ability 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.

In this book Chitta Baral proposes to use the non-monotonic language of AnsProlog – logic programming with answer set semantics, for both knowledge representation and reasoning, and declarative problem solving. He presents results obtained over the years in a first ever compilation. This compilation is not only unique to logic programming, but is unique to the field of knowledge representation as a whole, as for no other language or logic a comparable body of results has been accumulated. The book is targeted towards students, researchers and practitioners and its content includes: (a) theoretical results about AnsProlog such as, properties of AnsProlog sub-classes including their complexity and expressibility, and building block results to both analyze and modularly build large AnsProlog programs; (b) illustrations of correctness proofs of AnsProlog programs; (c) algorithms to compute answer sets ('models') of AnsProlog programs; (d) demonstration of the knowledge representation and reasoning ability of AnsProlog through benchmark examples such as the frame problem, reasoning about actions, inheritance hierarchies, reasoning with priorities, and reasoning with incomplete information; (e) use of AnsProlog in problem solving activities such as planning, diagnosis, constraint satisfaction problems, logic puzzles, and combinatorial auctions; and (f) descriptions of systems that implement AnsProlog and code of various examples in the syntax of these systems.

About the author

Chitta Baral is an associate professor at the Arizona State University. He obtained his B.Tech(Hons) degree from the Indian Institute of Technology, Kharagpur and his M.S and Ph.D degrees from the University of Maryland at College Park. He has been working in the field of knowledge representation and logic programming since 1988, and his research has been supported over the years by National Science Foundation, NASA, and United Space Alliance. He received the NSF CAREER award in 1995 and led successful teams to AAAI 96 and 97 robot contests. He has published more than 65 articles in Logic Programming, Knowledge Representation, and Artificial Intelligence conferences and Journals.

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 non-monotonic languages, and with the Prolog programming language. We then present the syntax and semantics various sub-classes of AnsProlog, and consider two views of AnsProlog programs: stand alone programs, and functions. We present 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/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, functional, compilable to first-order theory, amenable to removal of **or**, amenable to computation by Prolog, and restricted monotonicity – exhibition of monotonicity with respect to a select set of literals.

  We also define several subclasses 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 with 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 in 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-colarability, 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 do 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 problem of knowledge representation and reasoning formalisms. We show how AnsProlog is up to this task. Reasoning about actions also form the ground work for planning with actions, an important problem solving task. We present AnsProlog encodings of planning such that the answer sets each encode a plan. (ii) Our second motivation is in regards 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 later sections we consider more expressive scenarios.

- **Chapter 6: Complexity, expressibility and other properties of AnsProlog programs**
  In this chapter we consider some broader properties that help answer questions such as: (a) how difficult it is 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 expressibility, and present definitions of the polynomial, arithmetic and analytical hierarchy and their normal forms. We later use them in showing the complexity and expressibility 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 the 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. Finally, we briefly mention some of the other systems that can accept particular sub-classes of AnsProlog programs.

- **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 epistemic operators, and doing abductive reasoning. We also discuss some of the alternative characterizations of programs in AnsProlog syntax.

- **Appendices**
  We have several appendices: some of them in the book, and others in a companion web site. In the appendices in the book we present some background information, such as definition of ordinals, lattices, fixpoints, and definitions of Turing machine. In the web site we have Smodels and dlv code of several programs discussed throughout the book and also a list of pointers to resources such as home pages of active scientists and researchers in this field, and web pages of implemented systems.

Contents