Review of the book Answer Set Programming *by Vladimir Lifschitz*

Victor W. MAREK
University of Kentucky, Lexington, KY 40506, USA
(*e-mail:* `marek@cs.uky.edu`)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

There are at least two different interpretations of the title of this book: *Answer Set Programming* (ASP). One of this is the name of a topic which is part of the area of Computer Science called *Knowledge Representation* that deals with the techniques for representation of knowledge in a manner that can be used by computers. But there is yet another way that one can think about *Answer Set Programming*, namely as a form of *Constraint Programming*. This interpretation of ASP is what this book is about. The author, a prominent investigator of the ASP in the first meaning of the term decided to make an excursion in the use of it in the second sense. As such, the ASP forms an underpinning of the tool for *solving constraints*. The author did not treat this task as a theoretical venture; he *taught* an undergraduate course based on this idea. This had consequences; a large number of examples and exercises are provided in the body of the book. To make this book even more valuable, an effort has been made to provide *solutions* to the exercises. An appendix with these takes, roughly, 15% of the book. We need to stress that *all* exercises are solved, not just selected ones. This makes it possible for a student to use the text in self-study. Moreover solutions often have comments that explain *how* and *why* of the process of solving.

The book itself consists of preface, nine chapters, conclusions, an appendix with the solutions of problems and, finally, an extensive bibliography of over one hundred and twenty entries devoted to the eponymous *Answer Set Programming*, in the first of the above meanings. The roots of ASP as a formalism that can be viewed as related to Knowledge Representation, Database Theory, Constraint Solving, Planning, and a number of other areas of Computer Science are visible in various places of the book. Turning to the second meaning of the term, on a practical level, ASP is supported by *solvers* that provide solutions to problems formulated in the ASP formalism.

The book, as stated in Chapter 1, *Introduction*, grew out of the classes that the author of the book taught at the University of Texas, Austin. The class itself (and it shows in the text of the book) was part of their undergraduate program, which led to the elimination of various deeper aspects of ASP (for instance, complexity issues for ASP).

The idea of teaching such course at a relatively low level led the author of the book to tie the lecture to the specific computer system called *CLINGO*, designed and implemented at the University of Potsdam, Germany. That choice — likely to make the matters easier for the audience — has its consequences. Future significant changes in CLINGO, and especially in its language (and the history of computing and especially Information *Technology* in particular, is full of such situations) may make the book not adequate to the available ASP systems. The author does not address this issue, but the individuals teaching ASP (in the second sense of the concept) will need to adjust their courses correspondingly. But this “complaint” may be irrelevant — the presence the book itself, its clarity and the effort made to ensure that the students (in this case undergraduates) learn the subtleties of the language of CLINGO (via the above-mentioned extensive collection of exercises) may influence designers of future ASP systems to limit the changes in the syntax of the language of CLINGO.

Now, to the contents of the book. Chapter 1, *Introduction*, discusses the history of the subject and refers to the past attempts to use ASP as a constraint solving formalism.

Chapter 2, *Input Language of CLINGO*, introduces the reader to the language used by the system CLINGO. The language allows both symbolic computation, and arithmetical computation (since the vocabulary of the language allows both kinds of expressions). Likely, the most important aspect of this chapter is the introduction of *choice rules* which are responsible for the multiple solutions to problems (the solutions are expressed as so-called *answer sets*).

Chapter 3, *Combinatorial Search*, introduces the CLINGO system as a tool for solving constraints in the context of graphs. Numerous examples of problems in the class NP that can be presented in the CLINGO language and often solved by the CLINGO system are discussed.

Chapter 4, *Propositional Programs and Minimal Models*, introduces the reader to the semantics of CLINGO programs, and, in particular, minimal models. The material covered here includes a quick introduction to classical logic. Definite programs are introduced as well as their least model. A bit surprising is the lack of reference to the theoretical basis of the existence of the least model (Knaster-Tarski fixpoint theorem) and its algorithmic expression in finite case, Dowling-Gallier linear complexity algorithm. The technique of *Propositional Image*, a basic tool for handling semantics of CLINGO programs is introduced here.

Chapter 5, *Programs with Negation*, introduces the reader to the programs with negation and their *stable models*. The technique of Propositional Images is used here. However, the original approach of the author introduced with his collaborator, Michael Gelfond, (see the entry [1] of the bibliography) is — in the opinion of this reviewer — more intuitive and ties to elegant mathematical techniques of Universal Algebra. The same chapter brings in one of the fundamental results of Answer Set Programming, called in the text “Theorem on Constraints” and closely related to what researchers of Knowledge Representation call “Confirmation of the Evidence”.

Chapter 6, *Mathematics of Stable Models*, introduces the concept of *Strong Equivalence* of programs, the property where the equivalence (having same stable models) is preserved under union with other programs. Other concepts introduced in that chapter include the concept of the *completion of the program*, an older concept that is useful in the process of computation of stable models. The results in this chapter, are not, unfortunately, provided with proofs.

Chapter 7, *More on the Language of CLINGO*, discusses the aggregate constructs available in the language of CLINGO, including maximum, minimum, and summation. The presence of these constructs points to some of the roots of ASP, where one of the motivations of ASP was information extraction from relational databases. Some historical aspects of ASP, for instance, the construct of *classical negation* are also discussed in this chapter.

Chapter 8, *Dynamical Systems*, relates to handling reasoning about *effects of actions*. Classical aspects of systems admitting actions (for instance systems describing behavior of devices that have states that may evolve under actions from the outside of the system (industrial robots are examples of those) are discussed here. The natural application to such systems is in *planning*, *prediction* and *concurrency*, all briefly discussed in this chapter.

Short Chapter 9 provides conclusions of the book.

As mentioned above a very extensive collection of answers to exercises is provided in *Appendix A*. While this reviewer looked at only few solutions in the Appendix, those inspected are