Book review

Thinking as Computation: a first course by Hector J. Levesque, xx + 299 pages, The MIT Press, 2012. Hardcover, ISBN 978-0-262-01699-5.

The book is about thinking understood as a computational process and is intended as a textbook for undergraduate students who are interested in the connection between ordinary thinking as performed by people and computing.

The book may also be seen as an introduction to Prolog programming language or as an introduction to artificial intelligence. Thus it may be of interest to computer science students, as well.

Two chapters, the first and the last one, discuss some philosophical problems concerning thinking and computation. The remaining 11 chapters introduce Prolog programming language and present successively more complex problems showing how to solve them in Prolog. Most of those chapters have a similar general structure: a new kind of problems (described as requiring a new sort of thinking) is introduced first and subsequently a general way of solving such problems is discussed. A few example problems of that kind are then presented and Prolog programs for solving them are given. In total more than 50 Prolog programs are presented in the book.

Chapters 2–4 and 7 form a gentle introduction to Prolog. Chapter 2 informally presents Prolog clauses as simple sentences of two kinds: facts and implications. Back-chaining computational procedure is already introduced there. Chapter 3 is a formalization of the previous one. Moreover, a simple form of unification, negation and tracing of Prolog programs are also presented. Chapter 4 introduces recursion in Prolog and discusses soundness and completeness of Prolog programs (without using such notions). Problems of termination and efficiency of Prolog programs are also informally sketched. Lists in Prolog are presented in Chapter 7.

The remaining chapters (5, 6, 8–10) focus on selected kinds of problems. The idea of a constraint satisfaction problem is introduced in Chapter 5 (constraint optimization problems are not considered). The simplest form of solving such problems, i.e. generate-and-test technique is presented and the exponential growth of the search space is discussed. The idea of constraints is used also in Chapter 6 which deals with object recognition and interpretation of visual scenes.

Chapter 8 deals with the subject of understanding natural language. Notions of a grammar and parsing are introduced, problem of ambiguity of grammars is discussed and a few simple parsers in Prolog are developed. Definite Clause Grammars could be very useful here but they are not used. All presented Prolog programs use `append` predicate to nondeterministically split an analyzed sentence into proper parts. Such programs are simple and easy to understand (an important issue in the book) but their efficiency is rather poor (this topic is not discussed).

Planning problems are introduced in Chapter 9. Depth-first search and best-first search procedures are presented. A new representation of states (situations and fluents) suitable for bigger dynamically changing worlds is also presented.

Chapter 10 deals with playing strategic games. A representation of a game, its states and moves is presented first and subsequently the problem of finding a best move in a game situation is elaborated upon. The important problem of efficiency is discussed next and the sizes of the game trees for a few popular games are estimated. Finally, more complex search algorithms, such as the minimax algorithm and the alpha-beta pruning are presented.

The back-chaining procedure presented in Chapter 2 is used throughout the book. Other forms of reasoning are briefly mentioned in Chapter 11, including deduction, abduction and induction. A few extensions of the back-chaining procedure and the forward-chaining procedure are also presented. The chapter ends with presentation of a propositional calculus (CNF formulas) and barely mentions the first-order logic.

Each chapter ends with a section entitled *Want to read more?* containing not only an extensive guide to literature but also quite interesting short notes on the history of philosophy and computation.

Throughout the book a nontechnical approach is used. The presentation of Prolog programming language should be understandable by students without mathematical background, whereas many textbooks on Prolog are dedicated to computer science students only. Some notions well-known to computer science students (such as finite automata, for instance) are presented and used only informally. The book shows that it seems possible to teach programming in Prolog students without a technical background, without teaching them how to program in general.

Many programming features of Prolog are presented in the book. It is worthwhile to notice that each feature is presented only when it is needed to solve another problem. Thus for instance, negation is introduced already in Chapter 3, arithmetic and anonymous variables in Chapter 5, lists are presented only in Chapter 7 and dynamic predicates in Chapter 8.

The power of declarativeness of Prolog is showed using a few examples. For instance it is showed that parsers presented in Chapter 8 could be used not only to analyze given phrases but also to generate some missing parts of such phrases, or even to generate whole phrases referring to a given object.

The problem of efficiency is discussed in a few sections. Using simple but quite interesting examples it is showed what computers could do and what the limitations are. Thus the crushing power of combinatorics is revealed. Moreover it is showed that in some cases better algorithms are needed and different kinds of solutions have to be developed (e.g., a decomposition of a problem, other search algorithms, a new representation of states).

The main philosophical idea that reccurs throughout the book is that (simple kinds of) thinking can be regarded as a computation. Probably everyone would agree that thinking required to solve simple problems (e.g., puzzles) is a form of computation. But claiming that a program solving a puzzle or a planning problem does think is an exaggeration and anthropomorphism. So some statements present in the book are, to say the least, controversial.

The book presents only selected features of Prolog. Lack of some Prolog features or simplifications of others is by all means a meaningful decision. Nevertheless, there are a few mistakes. For instance, the description of the built-in predicate `is/2` and some comments on the arithmetic in Prolog are incorrect. A simplified usage of `is/2` could be presented, but not the incorrect one.

The problem of renaming variables is dealt with in a few sections. At first it is correctly stated that “variables in the sentences of a knowledge base should be renamed”, but later on one can also read that “Prolog renames the variables in a query”, which is incorrect. Regrettably, such a renaming of variables in queries is done also in a few examples.

Function symbols are used just in one section. Unfortunately, compound terms are incorrectly called atoms. Both predicate and function symbols (including constants) are called predicates.

There is also a statement which may mislead the readers. The author claims that forward-chaining procedure will never get stuck in a loop. This is not true in general. It could be regarded as true due to restrictions which are implicitly used in that section. But that should be stated explicitely.

To summarize, the book is an interesting introduction to many fields, including thinking, computation and programming in Prolog. The presentation is nontechnical, yet it is quite rigorous. Students of philosophy or linguistics, as well as students of computer science, can find interesting material in the book. Reading it will be a real pleasure to many readers.

Mirosława Miłkowska
University of Warsaw, Poland