Book review

(written for *Artificial Intelligence in Medicine*)

Kim Marriott, Peter J. Stuckey
Programming with Constraints : An Introduction
The MIT Press, 1998
ISBN number: 0262133415
467 + xiv pages, hardcover
price: $45.00

Historical background

Let me start this book review with a short historical perspective. As the authors state in the opening sentence of their book "Constraint programming is one of the most exciting developments in programming languages in the last decade". In this approach the programming process is limited to a generation of constraints and a solution of them by domain specific and general methods.

This approach to programming is an outcome of a research that originated within Artificial Intelligence in the early seventies in the works that dealt with computer vision. This led to an investigation of constraint satisfaction problems.

In the eighties the logic programming paradigm was extended by employing constraint solving instead of the unification to assign values to variables. This led to constraint logic programming, at this stage the most successful approach to constraint programming. Various constraint logic programming systems usually employ constraint satisfaction techniques in the form of some hard-wired methods transparent to the user or in the form of various built-ins.

More recently, the area of constraint programming profited from the link to Operations Research, when dealing with various optimization problems, in particular linear and integer programming problems. In the last ten years constraint programming was successfully applied to a wide variety of domains, including interactive graphic systems, operations research problems, molecular biology, business applications, electrical engineering, numerical computing, natural language processing, software engineering, and computer algebra.

Yet, there are hardly any books on this subject. This naturally explains the need for the book under review.

Contents

This book is the first comprehensive book on constraint programming. It contains a wealth of material both on the theoretical foundations of the field and on the programming methodology and is likely to remain a standard reference in the field for a number of years. Here is a short table of its contents (taken from amazon.com):

In the first part of the book the authors introduce in Chapter 1 constraints as conjunctions of atomic formulas of a first-order language and constraint solvers as sound but possibly incomplete procedures that determine truth of a constraint in a given interpretation. The problem of solving a constraint is then defined as the task of finding an assignment that satisfies it over a given interpretation.

The authors illustrate the constraint solving process by means of two algorithms: Gauss-Jordan elimination and a variant of the Martelli-Montanari unification algorithm (called "a tree constraint solver").

In Chapter 2 various operations on constraints are studied, such as variable elimination and optimization (search for the best solution w.r.t. some objective function). The latter is illustrated by means of the Simplex algorithm that allows us to solve optimization problems on reals expressed by means of linear constraints and a linear objective function.

The final chapter of this part, Chapter 3, deals with specific constraint solving methods that concern constraints on finite domains. The main technique is a reduction of a given CSP to a smaller one obtained by reducing its domains or constraints while maintaining equivalence. For some reason, only various forms of domain reduction are considered here. In particular the notion of path-consistency is not mentioned. It is explained how these techniques can be incorporated into the backtracking search leading to a more efficient search process.

The second part of the book deals with constraint logic programming. The unusual and interesting decision of the authors was to introduce this topic without explaining the logic programming framework or Prolog first. Consequently, the presentation right from the beginning, in Chapter 4, focuses on the distinction between the "domain constraints" with the constraint solvers associated with them and the user defined constraints. This approach leads to a presentation of the computation mechanism in which unification is "delayed". To overcome the often resulting complex and cumbersome goals so-called simplified derivation trees are introduced.

The subsequent chapters discuss this programming style in detail. The focus is on the modelling capabilities of constraint logic programming (in Chapter 5) and the use of various built-ins that are provided in constraint logic programming for declaring constraints and for guiding the search.

The presentation is clear and intuitive with the pedagogically sound idea of presenting the cut, the "goto" of logic programming, only as an implementation primitive and focusing instead on the "if_then_else" and "once" primitives.

The presentation in this part of the book intermixes an account of Prolog programming techniques, such as use of compound terms to represent lists, trees and records (in Chapter 6) with constraint programming techniques such as addition of redundant constraints (in Chapter 7).

Somewhat paradoxical is that the logic programming modelling of the iteration by means of tail recursion, well-known in the imperative programming style, is now explained (in Chapter 5) by means of one of the early jewels of constraint logic programming, the mortgage program.

Chapter 8 goes into detail into constraint programming over finite domains. Various primitives and programming techniques that deal with search are introduced. In my view especially relevant is Section 8.4, entitled "Different Problem Modelling" that makes it clear that (see page 269) "Determining the relative efficiency of different models is a difficult problem and one which relies upon an understanding of the underlying constraint solver" (in short: constraint programming [on finite domains] is more an art than a science).

In turn, in Chapter 9 it is explained how the constraint solvers can be extended or specialized to user defined constraints. Finally, in Chapter 10 it is discussed how (and why) the constraint solvers can be modified to become incremental, and selective implementation issues, such as management of the choice points, are explained.

The final part of the book is devoted to a short presentation of alternative approaches to constraint programming. This includes constraint databases (in Chapter 11), which amount to constraint logic programs executed in the bottom up fashion (to compute all answers to a goal), and other constraint programming languages (in Chapter 12) in which a variety of alternative programming languages that fall into this category are briefly surveyed, including programming systems that use libraries (such as the ILOG solver) and "mathematical languages" that allow us to handle constraints by means of packages, such as Mathematica.

Presentation

So much about the summary of the book contents. Now some comments about the writing style. As the authors state in the preface, this book is meant for (I assume computer science) students of fourth year. The authors clearly succumbed to the omnipresent increasing pressure to make the material "easily accessible" to a wider audience, where the latter is translated into inclusion of many examples, incorporation of many general discussions, and deletion of the proofs of various claims.

Let me be more specific. Several examples and interesting case studies included in this book are extremely helpful when trying to master constraint (logic) programming and the authors should be congratulated with the rich and varied choice they made. In particular, the reader will profit a lot from such interesting case studies as computing a layout of a binary tree, designing a bridge, modelling options trading, and solving a scheduling problem. In all cases the expressiveness, versatility and ease of programming using constraint logic programming are amply explained.

On the other hand, numerous discussions and comments that accompany the presentation usually do not add anything intrinsically new and are of limited value, especially because they often consist of some general statements that usually are true, ... or perhaps not.

Let me illustrate it by three, almost arbitrarily chosen quotations out of which in my opinion only the second one is of any relevance.

page 261: "Choosing the variable with smallest domain limits the branching factor and potentially allows failure of the entire branch to be determined quickly, thus guiding the search to more profitable areas." (A more precise argument would refer to a smaller size of the resulting search space.)

page 286: "Given the somewhat complicated behaviour of CLP(FD) systems, it is often difficult for even the most experienced programmer to predict how a particular technique will affect the execution speed."

page 338: "No matter how high level the programming language, there comes a time when a seemingly correct program does not give the right answer and so must be debugged."

In such a presentation style it is difficult to draw a line between the topics that should be presented rigorously and those to be presented informally. In the book under review the presentation of the "general purpose" algorithms is the victim. Namely, these algorithms, written in a pidgin Algol, are regrettably introduced in an informal way, usually without a correctness proof (example: Bound Consistency Algorithm 3.6 on page 106), or even without a precise explanation of what the algorithm is meant to achieve (example: An incremental finite domain propagation-based solver on page 375, preceded by a vague remark "Thankfully [making consistency based solvers incremental] is quite simple because propagation is essentially an incremental process").

Such an ad hoc presentation of algorithms is regrettably common in Artificial Intelligence and amounts to correctness proofs by intimidation. In the case of constraint logic programs such a style is less offensive because these programs are closer to specifications.

A more rigorous presentation was certainly possible. For instance partial correctness of the unification algorithm is properly explained (though the not completely obvious proof of termination is relegated to the exercises) and in the informal explanation of the Incremental arithmetic constraint solver an appeal is made on page 381 to invariants.

The fact that this style is not followed in the rest of the book is an unavoidable consequence of the fact that correctness proofs of these algorithms rely on the proofs of various "technical" properties that were omitted. At one place such an informal presentation leads to an error: the definition of the negative derivation step on page 309 is incomplete, as can be witnessed by the goal p and the program p :- not(p). The correct definition is much more complex— see, e.g., Martelli and Tricomi, *Information Processing Letters 43(2)*, 1992.

Finally, a comment on constraint propagation. One would expect it to be a central topic for this book. Unfortunately, it is defined in a very informal way, usually by means of naive algorithms. In fact, even the classic arc consistency algorithm, `AC-3`, is not introduced and instead a very naive algorithm is given (on page 94).

Assessment

The outcome is a textbook that is somewhat unbalanced. Its strengths are impressive case studies and lucid presentation of the programming style. On the other hand, the book, while "locally" easy to read, is more difficult to grasp on a "global" level, because the crucial issues are often interleaved with not so relevant discussions and unsubstantiated remarks (such as "... in general, determining if an arbitrary primitive constraint is hyper-arc consistent is NP-hard" (a notion that is not explained) on page 97) and because the correctness (or even precise specification) of most of the algorithms is mercilessly left to the reader.

But one should also have in mind that this is the first comprehensive textbook on constraint programming. It maps a well defined area and it will be difficult in future to ignore the view of constraint programming it puts forward.

Krzysztof R. Apt
CWI
Amsterdam
November 25, 1998