Book review

Principles of Constraint Programming by Krzysztof R. Apt, Cambridge University Press, 2003, hard cover: ISBN 0-521-82583-0, xii + 407 pages, price: 50 US $ or 35 £. [1]

Constraint programming is a very active research area that lies at the intersection of AI, OR, Programming Languages, Data Bases, and Graph Theory, and whose main aim is to model naturally and solve efficiently real-life problems that can be cast as a set of objects and a set of restrictions (that is, constraints) on how these objects can co-exist in the same problem. Constraint programming started a long time ago, with the first propagation algorithms by Waltz, Montanari, Mackworth and Freuder in the late 70's, then it found high-level language support in Constraint Logic Programming (CLP) in the late 80's, and since then it has evolved to a very complex discipline with many different modeling languages and solving techniques.

K. R. Apt's book on constraint programming (Apt, 2003) has been published in 2003 and is thus one the latest books published on this subject. To be able to contextualize the value of this book, we first need to draw a brief history of the books that have been published on this same subject. The first book on constraint programming was published in 1989 (Hentenryck, 1989) by Pascal Van Hentenryck, and it showed how to exploit some of the constraint techniques typical of AI in the CLP world, by using the CHIP language. A few years later, in 1993, Edward Tsang wrote a book on constraint solving (Tsang, 1993), which contained the main notions as well as some advanced solving techniques based on local search and genetic algorithms. This book had an AI flavor, and did not deal with programming languages based on constraints such as CLP. In the same year, Vijay Saraswat published a book on Concurrent Constraint Programming (CCP) (Saraswat, 1993), a concurrent abstract model of computation, originated from extensions of CLP, where constraints are not only a way to express real-life restrictions, but also a way for concurrent agents to communicate. Van Hentenryck's book was a good source of information for CLPers, but as time passed new techniques, propagation algorithms, and language constructs were proposed, so the book on CLP by Marriott and Stuckey (Marriott & Stuckey, 1998), that appeared in 1998, updated the readers by containing the basic notions of constraint propagation, and by also giving a formal syntax and semantics of the current computational model of CLP. So, by 1998 people knew where to find the main notions and results of constraint solving, as well as the syntax and semantics of constraint programming languages like CLP and CCP. However, what was still missing was a place where to find also the most

[1] This book review was handled by Peter Stuckey.

advanced concepts and results, and also a formal and uniform way to look at the whole field.

In 2003 three new books on constraint programming came out, and they succeeded in solving both these problems. One is a very comprehensive book on most of the notions and results on constraint programming (Dechter, 2003), written by Rina Dechter, one of the main players in the field of constraint programming for more than 25 years, who exploited all her past work to make her book an invaluable source of information for both beginners and experienced CP researchers. The second one is a book by Thom Frühwirth and Slim Abdennadher (Frühwirth & Abdennadher, 2003), which focuses on recent advances on CLP based on Constraint Handling Rules, a high level formalism for defining solvers via concurrent rewriting rules. The third book is the book which is the subject of this review, whose main aim is to give a general and formal view of most of the techniques used in constraint programming.

K. R. Apt is a relative newcomer to constraint programming: after a lot of pioneering and essential work in program verification and logic programming, he started getting interested in the field of constraint programming around the mid 90's. A book on a subject written by somebody who has recently entered the subject can be one of two things: either a collection of naive statements and algorithmic descriptions, or a fundamental new way of looking at constraint programming. Apt's book falls in this second category.

The main objective of the book is two-fold. First, to give a formal proof-theoretic view of constraint propagation and search, the two main ingredients of constraint programming. Second, to use this view to define an abstract setting of function iterations over partial orders, which allows for stating very general results that can then be instantiated to the various algorithms for constraint propagation. In this way, he not only provides a uniform view of the existing techniques, but he also sets the ground for applications of this theory to any constraint-related technique that could be developed in the future. The abstract framework introduced in the book is very general, but it does not lack applicability. In fact, Apt succeeds in identifying the properties which are sufficient to make the framework usable in many practical cases. All this is done with the highest level of precision and formality, without however making the reading task too heavy, as it is typical of Apt's writing style.

The book consists of nine chapters. After a short introductory chapter, Chapter 2 provides a long list of examples of constraint problems, ranging from n-queens to cryptoarithmetic, and from temporal to Boolean constraints. For each example, variable, domains, and constraints are defined. This chapter is not too original in the list of chosen examples, considering that other constraint programming books start in a similar way. However, I cannot think of another reasonable way to start a book on constraints. Also, here the level of detail is much higher, especially in the description of temporal and qualitative constraints, and also in the analysis of polyhedral scenes.

Chapter 3 tries to convey in an informal way most the notions and results that will later be given formally in the rest of the book. This is done to allow the reader to see the overall picture. More precisely, constraint solving is described here as the

iterative application of preprocessing, constraint propagation, and problem splitting (into other problems where the overall procedure is then recursively applied), until we find a solution or the current problem is too small to be split. For each part of this procedure, the informal meaning is given, as well as some examples of its execution. This chapter is very important and original, since there is a big value in seeing the whole scenario rather than a separate description of all its ingredients, as most of the other books do. So I really appreciate the effort in giving an overview of what it means to solve a problem via constraint programming. There are some small drawbacks in doing this, but I guess they are unavoidable. In fact, there is some risk in giving a lot of information to the reader, and informally, at the beginning of the book, since this may create some confusion. For example, constraint propagation is defined here, in general, as a part of the solving procedure, but later in the book solvers are described in a way (that is, applications of domain-reduction or transformation rules) that matches the structure of the constraint propagation part, and not that of the overall solving procedure. So inexperienced readers should be warned (see description of Chapter 4 and 5). Moreover, some classes of constraints are defined here without giving all the details, and then later they are necessarily re-defined in a complete way. However, the level of precision of the book helps in avoiding getting confused after such a short introduction to constraint programming. So I really think this chapter, despite its informal character in a book which is mainly formal, has great value.

Chapter 4 focuses on examples of complete solvers. Here by a solver is meant the application of some proof rules until no rule is applicable or its application does not change anything. Completeness is meant to be the ability to generate a problem where it is easy to check if it is solvable or not. The main notions used in the rest of the book for describing constraint propagation are given here: proof rules, derivations, and problem equivalence. After such definitions, some examples of classes of proof rules, and the corresponding constraint classes for which they have been designed, is given, to show that they are complete. The first chosen example is the unification algorithm for term equations, for which a very clear and detailed description is given, that takes advantage of the exceptional logic programming background of the writer. Then there is a section on linear equations over reals with the Gauss-Jordan algorithm, where the only drawback is the need to adjust the previously given notions of substitution and unifier, and finally a section on linear inequalities over reals. These three examples of complete solvers are mainly useful, in my view, to get some familiarity with the proof-theoretic description of constraint propagation. It should now be clear to the reader that if a solver is complete, then the solving procedure of the previous chapter degenerates to just the constraint propagation phase; that is, no splitting is necessary.

Chapter 5 introduces the main local consistency notions: node, arc, and path consistency, and their directional versions, which are then all generalized to k-consistency and finally to relational consistency. For each notion, the corresponding proof rules to obtain such a level consistency are then defined. Especially useful is the presence of many examples to show the relation among all these notions in a problem. There is only one section devoted to results on tractability, and it refers

to the existing relation between the width of the graph representing a CSP and the level of local consistency which is necessary to assure the existence of a solution. It is a pity that no space has been found for other tractability results which can be found in the literature.

Chapter 6 shows some incomplete solvers: the classes of constraints considered are equality and disequality constraints, Boolean constraints, linear constraints on integer intervals (for which bound consistency is introduced), arithmetic constraints on integer intervals, and arithmetic constraints on reals. The amount of information and level of detail for each class is here at its highest degree, so that a reader interested in understanding fully what happens when using such solvers can find here all the information. For some classes, alternative solvers are also provided. As in Chapter 4, it should now be clear that incomplete solvers fit the overall solving procedure of Chapter 3 by instantiating only the constraint propagation part.

Chapter 7 describes constraint propagation algorithms as instances of some very general iteration algorithms. Such algorithms consider a set of functions and at each step apply one of such functions to an element of a partial order, starting from the bottom and stopping when stabilization is reached. At this point, the current element of the partial order is the least common fixpoint of the used functions. Three different iteration algorithms are introduced, with different features, which can be used when the functions have certain properties. If the functions commute and are monotonic and idempotent, then the first algorithm (just apply each of them once in any order) can be used; if they lack commutativity but have semi-commutativity w.r.t. a certain order of the functions, and moreover they are inflationary, then the second algorithm can be used (just apply each of them once in the chosen order). Otherwise, when the functions are not even semi-commutative, the third algorithm can be used: apply all the functions possibly repeating each of them, until a common fixpoint is reached. Although the description of the three algorithms and their properties is very abstract, it is very well written and easy to follow, and it is a joy to see how this machinery is then applied very easily to several known constraint propagation notions. After defining the partial order for CSPs and the functions to use, which are directly related to the proof-rules of the previous chapters, node consistency is shown to be able to use the first algorithm, directional consistency the second one, and k-consistency the third one. In my view, this is the best written and most original chapter of the book.

Chapter 8 is devoted to search algorithms. Starting from labeling trees where all complete instantiations are considered, smaller search trees are then introduced step by step, which represent the trees generated by using forward checking, partial look-ahead, and full-lookahead (MAC). After defining such trees, the search algorithms which build them are then defined, again starting from the simplest case of an algorithm which never backtracks to general backtracking algorithms which include constraint propagation at each tree node. Branch and bound is also defined to deal with constraint optimization. This incremental way of introducing complex search algorithms is original as far as I know, and gives an interesting new point of view on them.

The last chapter gives a brief overview of the main issues which have not found

space in the other chapters of the book but which have to be considered when using a constraint language or a solver: modeling, constraint languages (mainly constraint logic programming and ILOG solver), various issues about solvers like incrementality, other forms of search like limited discrepancy search and local search, over-constrained and soft constraint problems. By looking a this list, I wish Apt had devoted more space in the book (even a full chapter each) to at least two of these issues: local search and soft constraints. Local search is a way to look for solutions (or optimal solutions) which is completely different from the search approach considered in the book, and is gaining a lot of attention lately. So it would have been interesting to have a precise and formal account for it. As for soft constraints, I am biased having worked on this subject for long time, but since they can be easily cast in the abstract iteration framework of this book, they would be a nice additional proof of the generality of the framework. Also, a solving approach which is not considered in this book is the one based on variable elimination and dynamic programming, and which includes techniques like adaptive consistency and bucket elimination. It would have been interesting to see it described within the formal approach of this book.

Summarizing, even after having read several books on constraint programming, and having used some of them for my courses, I really enjoyed reading Apt's book and using it for my current course on constraint programming, mainly for two reasons. First, it is extremely well written (but this was predictable knowing Apt's writing style), so much that the formal style never makes reading a heavy task. Second, it provides a very original abstract view of the main concepts of constraint propagation and constraint programming, which can be very useful both for beginners who need to understanding these concepts without getting lost, and also for experienced researchers who may want to see where the current frontier for an abstract approach to constraint programming is and whether it can be moved further.

Moreover, for those who want to use the book in a course, Apt (as also the authors of some other books on constraint programming (Frühwirth & Abdennadher, 2003; Marriott & Stuckey, 1998)) has made wonderful Latex-generated pdf slides that cover all the material contained in the first eigth chapters of the book (see at the URL `http://homepages.cwi.nl/~apt/pcp/`). So you don't need to prepare teaching material if your students can cope with English slides.

Since the first time I met K. Apt in 1989, when I attended a Ph.D. course on logic programming that he was teaching, I have always been stunned by his exceptional ability to be formal and clear. This book is yet another witness for this ability, and it is a great present to the constraint programming community, which will certainly advance scientifically because of its publication. We need good books to educate new people to Constraint Programming in the best way, and this book is a great way to do this.

References

Apt, K. R. (2003). *Principles of constraint programming.* Cambridge University Press.

Dechter, R. (2003). *Constraint processing*. Morgan Kaufmann.

Frühwirth, T., & Abdennadher, S. (2003). *Essentials of constraint programming*. Springer.

Hentenryck, P. Van. (1989). *Constraint satisfaction in logic programming*. MIT Press.

Marriott, K., & Stuckey, P. J. (1998). *Programming with constraints: an introduction*. MIT Press.

Saraswat, V.A. (1993). *Concurrent constraint programming*. MIT Press.

Tsang, Edward P. K. (1993). *Foundations of constraint satisfaction*. Academic Press.

Francesca Rossi Dipartimento di Matematica Pura ed Applicata, Universitá di Padova, Padova,Italy