Principles of Constraint Programming

Krzysztof R. Apt

CWI, Amsterdam, The Netherlands

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011–4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2003

Printed in the United Kingdom at the University Press, Cambridge

Typeface Computer Modern 10/13pt *System* LaTeX 2ε

A catalogue record for this book is available from the British Library

ISBN 0 521 82583 0 hardback

Contents

1

Introduction

1.1 Basic characteristics of constraint programming

THIS BOOK IS about ***constraint programming***, an alternative approach to programming which relies on a combination of techniques that deal with ***reasoning*** and ***computing***. It has been successfully applied in a number of fields including molecular biology, electrical engineering, operations research and numerical analysis. The central notion is that of a constraint. Informally, a ***constraint*** on a sequence of variables is a relation on their domains. It can be viewed as a requirement that states which combinations of values from the variable domains are admitted. In turn, a ***constraint satisfaction problem*** consists of a finite set of constraints, each on a subsequence of a given sequence of variables.

To solve a given problem by means of constraint programming we first formulate it as a constraint satisfaction problem. To this end we

- introduce some variables ranging over specific domains and constraints over these variables;
- choose some language in which the constraints are expressed (usually a small subset of first-order logic).

This part of the problem solving is called ***modeling***. In general, more than one representation of a problem as a constraint satisfaction problem exists. Then to solve the chosen representation we use either

- domain specific methods,

or

- general methods,

or a combination of both.

The ***domain specific methods*** are usually provided in the form of implementations special purpose algorithms. Typical examples are:

- a program that solves systems of linear equations,
- a package for linear programming,
- an implementation of the unification algorithm, a cornerstone of automated theorem proving.

In turn, the ***general methods*** are concerned with the ways of reducing the search space and with specific ***search methods***. The algorithms that deal with the search space reduction are usually called ***constraint propagation algorithms***, though several other names have been often used. These algorithms maintain equivalence while simplifying the considered problem. They achieve various forms of ***local consistency*** that attempt to approximate the notion of (global) consistency. The (top down) search methods combine various forms of constraint propagation with the customary backtrack and branch and bound search.

The definition of constraint programming is so general that it embodies such diverse areas as Linear Algebra, Global Optimization, Linear and Integer Programming, etc. Therefore we should stress one essential point. If domain specific methods are available they should be applied *instead* of the general methods. For example, when dealing with systems of linear equations, the well-known linear algebra algorithms are readily available and it does not make sense to apply to these equations the general methods.

In fact, one of the aims of constraint programming is to look for efficient domain specific methods that can be used instead of the general methods and to incorporate them in a seamless way into a general framework. Such a framework usually supports

- domain specific methods by means of specialised packages, often called ***constraint solvers***,
- general methods by means of various built-ins that in particular ensure or facilitate the use of the appropriate constraint propagation algorithms and support various search methods.

Once we represent a problem as a constraint satisfaction problem we need to solve it. In practice we are interested in:

- determining whether the chosen representation has a solution (is consistent),
- finding a solution, respectively, all solutions,
- finding an optimal solution, respectively, all optimal solutions w.r.t. some quality measure.

After this short preview we can formulate the following basic characteristics of constraint programming:

Two Phases Approach: The programming process consists of two phases: a generation of a problem representation by means of constraints and a solution of it. In practice, both phases consist of several smaller steps that can be interleaved.

Flexibility: The representation of a problem by means of constraints is very flexible because the constraints can be added, removed or modified. This flexibility is inherited by constraint programming.

Presence of Built-ins: To support this approach to programming several built-in methods are available. They deal with specific constraint solvers, constraint propagation algorithms and search methods.

An additional aspect brought in by constraint programming is that modeling by means of constraints leads to a representation of a problem by means of relations. This bears some resemblance to database systems, for instance relational databases. In fact, constraints are also studied in the context of database systems. They are useful in situations where some information, for instance the definition of a region of a map, needs to be provided implicitly, by means of constraints on reals.

The difference is that in the context of database systems the task consists of efficiently querying the considered relations, independently on whether they are defined explicitly (for instance by means of tables) or implicitly (for example by means of recursion or inequalities). In contrast, in constraint programming the considered relations are usually defined implicitly and the task consists of solving them or determining that no solution exists. This leads to different methods and different techniques.

1.2 Applications of constraint programming

Problems that can be best solved by means of constraint programming are usually those that can be naturally formulated in terms of requirements, general properties, or laws, and for which domain specific methods lead to overly complex formalisations. Constraint programming has already been successfully applied in numerous domains including:

- interactive graphic systems (to express geometric coherence in the case of scene analysis),
- operations research problems (various optimization problems, in particular scheduling problems),
- molecular biology (DNA sequencing, construction of 3D models of proteins),
- business applications (option trading),
- electrical engineering (location of faults in the circuits, computing the circuit layouts, testing and verification of the design),
- numerical computation (solving polynomial constraints with guaranteed precision),
- natural language processing (construction of efficient parsers),
- computer algebra (solving and/or simplifying equations over various algebraic structures).

More recent applications of constraints involve generation of coherent music radio programs, software engineering applications (design recovery and code optimization) and selection and scheduling of observations performed by satellites. Also, constraint programming proved itself a viable approach to tackle certain computationally intractable problems.

While an account of most of these applications cannot be fit into an introductory book, like this one, an interested reader can easily study the research papers on the above topics, after having acquainted himself/herself with the methods explained in this book.

The growing importance of this area can be witnessed by the fact that there are now annual conferences and workshops on constraint programming and its applications that consistently attract more than one hundred (occasionally two hundred) participants. Further, in 1996 an (unfortunately expensive) journal called ‘Constraints’ was launched. Also, several special issues of computer science journals devoted to the subject of constraints have appeared. But the field is still young and only a couple of books on this subject have appeared so far. This led us to writing this book.

1.3 A very short history of the subject

Before we engage in our presentation of constraint programming, let us briefly summarise the history of this subject. It will allow us to better understand the direction the field is heading.

The concept of a constraint was used already in 1963 in an early work of I. Sutherland on an interactive drawing system SKETCHPAD. In the seventies various experimental languages were proposed that used the notion of constraints and relied on the concept of constraint solving.

The concept of a constraint satisfaction problem was also formulated in the seventies by researchers in the artificial intelligence (AI). They also identified the main notions of local consistency and the algorithms that allow us to achieve them. Independently, various search methods were defined. Some of them, like backtracking can be traced back to the nineteenth century, while others, like branch and bound, were defined in the context of combinatorial optimization. The contribution of constraint programming was to identify various new forms of search that combine the known techniques with various constraint propagation algorithms. Some specific combinations were already studied in the area of combinatorial optimization.

In the eighties the first constraint programming languages of importance were proposed and implemented. The most significant were the languages based on the logic programming paradigm. This led to a development of ***constraint logic programming***, an extension of logic programming by the notion of constraints. The programming view that emerged led to an identification of ***constraint store*** as a central concept. Constraint propagation and various forms of search are usually available in these languages in the form of built-ins.

In the late eighties and the nineties a form of synthesis between these two developments took place. The researchers found various new applications of constraint programming, most notably in the fields of operations research and numerical analysis. The progress was often achieved by identifying important new types of constraints and new constraint propagation algorithms. One also realised that further progress may depend on a combination of techniques from AI, operations research, computer algebra and mathematical logic. This turned constraint programming into an interesting hybrid area, in which theoretical work is often driven by applications and in turn applications lead to new challenges concerning implementations of constraint programming.

1.4 Our approach

In our presentation of the basic concepts and techniques of constraint programming we strive at a streamlined presentation in which we clarify the nature of these techniques and their interrelationship. To this end we organised the presentation around a number of simple principles.

Principle 1: Constraint programming is about a formulation of the problem as a constraint satisfaction problem and about solving it by means of domain specific or general methods.

This explains our focus on the constraint satisfaction problems and constraint solvers.

Principle 2: Many constraint solvers can be naturally explained using a rule-based framework. The constraint solver consists then of a set of rules that specify its behaviour and a scheduler. This viewpoint stresses the connections between rule-based programming and constraint programming.

This explain our decision to specify the constraint solvers by means of proof rules that transform constraint satisfaction problems.

Principle 3: The constraint propagation algorithms can be naturally explained as instances of simple generic iteration algorithms.

This view allows us to clarify the nature of the constraint propagation algorithms. Also, it provides us with a natural method for implementing the discussed constraint solvers, since a rule scheduler is just another instance of a generic iteration algorithm.

Principle 4: (Top down) search techniques can be conceptually viewed as traversal algorithms of the search trees.

This explains why we organised the chapter on search around the slogan:

Search Algorithm = Search Tree + Traversal Algorithm,

and why we explained the resulting algorithms in the form of successive reformulations.

1.5 Organisation of the book

The above explained principles lead to a natural organisation of the material. Here is a short preview of the remaining chapters. In **Chapter 2** we discuss several examples of constraint satisfaction problems. We stress there that in many situations several natural representations are possible. In **Chapter 3** we introduce a general framework that allows us to explain the basics of constraints programming. We identify there natural ingredients of this framework. This makes it easier to understand the subject of the subsequent chapters.

Then, in **Chapter 4**, we provide three well-known examples of complete constraint solvers. They deal, respectively, with solving equations over terms, linear equations over reals and linear inequalities over reals. In turn, in **Chapter 5** we introduce several notions of local consistency and characterise them in the form of proof rules. These notions allow us to study in **Chapter 6** in more detail a number of incomplete constraint solvers that involve Boolean constraints and linear and arithmetic constraints on integers and reals.

In **Chapter 7** we study the constraint propagation algorithms that allow us achieve the forms of local consistency discussed in Chapter 5. The characterisation of these notions in the form of proof rules allows us to provide a uniform presentation of these algorithms as instances of simple generic iteration algorithms. Next, in **Chapter 8**, we discuss various (top down) search algorithms. We present them in such a way that one can see how these algorithms are related to each other. Finally, in **Chapter 9**, we provide a short overview of the research directions in constraint programming.

Those interested in using this book for teaching may find it helpful to use the transparencies that can be downloaded from the following website: `http://www.cwi.nl/~apt/pcp`.