Book Review
(written for JOLLI)

Book title: *Essentials of Constraint Programming.*
Authors: *Thom Frühwirth, Slim Abdennadher.*
Publishing company: *Springer.*
Series: *Cognitive technologies.*
ISSN: 1611-2482, ISBN: *3-540-67623-6.*
Publication year: *2003.*
N. of pp.: 144.

Introduction

Constraint programming is a growing area of research, which involves people from various fields: artificial intelligence, computational linguistics, computational logic, computer algebra, operations research and many others. Since 1995, it features a yearly conference (*Principles and Practice of Constraint Programming*) which attracts contributions from all the aforementioned fields.

As the authors clearly state in the Introduction to their book, the underlying idea of constraint programming is rather simple in itself: "to solve problems by simply stating constraints (conditions, properties) which must be satisfied by a solution of the problem". More in details, this process first involves modelling the problem by means of a set of variables, each ranging on a specific domain, and constraints to restrict the variables' domain values. Constraint problems can then be specified and solved by means of a constraint programming language; once placed in the so-called 'constraint store', constraints trigger the appropriate constraint solvers, which may be interleaved with search. For instance, born in the 1980's as an extension of logic programming, constraint logic programming is still one of the most popular approaches to constraint programming; functional, imperative and object-oriented constraint languages are also well developed.

The first book on constraint programming appeared in 1998; this is a comprehensive and voluminous book, mainly intended for undegraduate students. Instead, *Essentials of Constraint Programming* stands on a different level; addressed to "graduate students, researchers and practitioners in [...] computer science and related fields" with a solid theoretical background, the book distinguishes itself as a concise, formal introduction to the field of constraint programming.

Contents

One of the main characteristics of this book is disclosed by the chosen title: essentiality. In general, concision and precision come in pairs in this book, and the authors should be congratulated for this. If the title perfectly mirrors the writing style (and we will come back on this below), a note of warning is in order with respect to the scope and contents of this book: this mainly focuses on (concurrent) constraint logic programming, and constraint handling rules — a concurrent constraint programming language. So statements such as "the book is a short, concise and complete presentation of constraint programming" (book cover and preface) are better understood as the main focus of the book is made clear. Here is a short table of contents to help the reader assess the book contents:

```
1.  Introduction
Part I. Constraint Programming
2.  Algorithm = Logic + Control
3.  Preliminaries of Syntax and Semantics
4.  Logic Programming
5.  Constraint Logic Programming
6.  Concurrent Constraint Logic Programming
7.  Constraint Handling Rules
Part II. Constraint Systems
8.  Constraint Systems and Constraint Solvers
9.  Boolean Algebra B
10.  Rational Trees RT
11.  Linear Polynomial Equations
12.  Finite Domains FD
13.  Non-linear Equations I
Part III. Applications
14.  Market Overview
15.  Optimal Sender Placement for Wireless Communication
16.  The Munich Rent Advisor
17.  University Course Timetabling
Part IV. Appendix
A. Foundations from Logic
```

The book is composed of three parts (quoting from p. 2): "the first part dicusses classes of constraint programming languages"; "the second part introduces types of constraints and algorithms" to solve constraint problems; "the third part describes three exemplary applications in some detail". The appendix surveys the basics of first-order logic as it is necessary for the comprehension of this book; first-order languages are used as formal specification languages, and first-order logic gives the declarative semantics of the presented (logic) programming languages. Each of this part is described in more details below.

First part. The opening of this part is by the title of a seminal paper by Robert Kowalski: *Algorithm = Logic + Control* (CACM, 1979). This introduces the surveyed programming languages: logic programming; constraint logic programming; concurrent constraint logic programming; constraint handling rules. Of all these languages, the authors give the syntax, declarative and operational semantics, briefly summing up the main results concerning correctness and completeness. The outcome is a uniform presentation, and the chosen subdivision of the material reflects the historical evolution of those constraint programming languages. Each language is introduced by a concise yet precise historical table, which displays the main steps in the historical evolution of the language itself (except for constraint handling rules).

More in details: after presenting the common notations for the syntax and semantics of such languages (Chapter 2), in Chapter 3 the authors introduce logic programming as "the basis of most constraint programming languages" (p. 14). Given this central role to logic programming, the authors spend a certain number of pages on this language; the *don't-know* non-deterministic search of logic programming is well discussed and accompanied by clarifying examples. Constraint logic programming is then introduced as a "natural combination of two declarative paradigms: constraint solving and logic programming". Here is another example of the essentiality of this book: at the end of the 1970's "efforts were made to make logic programming (LP) more declarative (with a flexible selection strategy), faster (with improved search), and more general (with extended equality)". This sentence gives in essence the key to understanding the revolution brought forward by adding constraints to logic programming. At this point in the book, there is not much discussion on the extended modelling capabilities or the "improved search" of constraint programming; the reader is referred to the second part of the book for this. Concurrent constraint logic programming (CLP) languages are briefly presented in Chapter 6 as committed-choice languages: the *don't know* non-deterministic search of (constraint) logic programming is here replaced by a *don't care* non-deterministic search. This allows the authors to smoothly introduce the language of constraint handling rules (CHR): "CHR is essentially a concurrent committed-choice language", that was originally designed for writing constraint solvers and evolved as a general purpose concurrent constraint language. All the constraint solvers and applications in the remainder of the book are written in CHR or $\text{CHR}^{\vee}$, which extends

CHR by allowing disjunctions in the right-hand side of rules to subsume the *don't know* non-determinism of CLP. One would then expect a complete introduction to CHR and CHR$^\vee$; but the authors prefer to keep their presentation compact, even though sometimes a more extensive presentation would be beneficial. For instance, there is not much discussion on the role of disjunctions in CHR$^\vee$ in this part (though there are clarifying examples of their use in Part II of this book), nor are any results on the soundness and completeness of this extension of CHR presented. Anyway, the presentation is sufficient for the comprehension of the remainder and there are references for further readings on the topic.

Second part. After introducing a general framework for constraint programming systems (Chapter 8), the authors survey a number of relevant constraint solvers in constraint programming, namely solvers for: boolean constraint problems; rational trees; linear polynomial equations over real numbers; constraint satisfaction problems over finite domains; non linear equations over real numbers. CHR or CHR$^\vee$ are the chosen languages for writing (parts of) these solvers; each of these is presented as an instance of the general framework of Chapter 8, and each comes with some exemplary applications which helps assess its use. The result is again a uniform presentation, which smoothly follows from the first part of this book.

A note on the generality of this presentation is in order here. The "allowed constraints" in a constraint system are presented as "a set of [first-order] formulae that contains the constraints true, false and $\doteq$, and that is closed under existential quantification and conjunction" (p. 53). This is not the most common approach to the modelling of a problem as a constraint problem, but it perfectly fits within the scope of this book, which focuses on logic-based languages. However, this choice affects the presentation of the 'local consistency' notions in Chapter 12; these are well-known properties of constraint (satisfaction) problems over finite domains, among which arc consistency is the most popular. Its presentation in terms of the adopted allowed constraints obscures its generality. Also, Section 12.1 goes under the name of arc consistency; but what is therein introduced is commonly regarded as domain propagation (or reduction) of constraint problems, which encompasses arc consistency and bounds consistency. Also, a more extensive introduction to search in (logic-based) constraint programming would be a valuable add-on; a number of search algorithms beyond and improving on "backtracking + local consistency algorithms" are nowadays implemented in most constraint programming languages and systems. However, the interested reader can already find a number of research papers and books on this topic, mainly in the artificial intelligence literature.

Third part. The third part of this book is about successful commercial applications of constraint programming. All the applications are briefly and clearly presented via CHR; as put forward by the authors, the code of these applications is short and easy to maintain.

More information on such and similar applications is available at `http://www.informatik.uni-ulm.de/pm/mitarbeiter/fruehwirth/`. The linguistic applications of CHR there presented will sound appealing to the more linguistics-inclined readers, that nowadays can be counted in the numbers in computer-science related disciplines. CHR and more in general constraint logic programming feature a number of applications in theorem proving, also beyond first-order logic: e.g., basic description logics. So the reader of this book should make sure not to miss the web pages of CHR for information on other more academic but equally interesting applications.

Presentation

As the authors put it, "this book is ideally suited as a textbook for graduate students and as a resource for researchers and practitioners". Thus the book requires a certain degree of 'acquaintance' with formal methods to be read. However the uniform style of this book makes it easier and easier to be read. Its structure is also well designed and concocts to create a fluid presentation: for instance, the subdivision in three parts clearly separates (logic-based) constraint programming languages from constraint systems, and their applications. Also, constraint programming languages are presented in a uniform manner: first is a short historical introduction, then come the syntax, the declarative and operational semantics of the languages; this helps catch, at a glance, their commonalities and differences.

Through all their book, the authors make an effort to be essential and their effort results in a rigorous and concise presentation. Sometimes, this comes with a price and some minor slips intrude: e.g., Definition 3.0.1 on p. 10 requires to know what the reflexive and transitive closure of a relation is; but, some lines below on that page, the authors take some space to define equivalence relations, thus reflexivity and transitivity get introduced at this point. Also, the definition of congruence is somehow missing (p. 10); indeed a congruence is also an equivalence relation, but what makes an equivalence relation worth the name of congruence is not further specified.

The addition of more references to relevant work in the literature would also be beneficial (e.g., the acronym SLD and breadth-first search are quickly introduced on p. 17 but never explained elsewhere). However, a manual of artificial intelligence may be sufficient and prove to be a good companion to this book.

Assessment

The book results in a uniform introduction to constraint programming, focused on logic-based programming. Anybody looking for a formal, essential but never-shallow introduction to the field should definitely consider this book.

Rosella Gennari
SRA ITC-irst
Trento, 16 February 2004