

# 1. Introduction

*Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.*

Eugene C. Freuder, Inaugural issue of the *Constraints Journal*, 1997

The idea of constraint-based programming is to solve problems by simply stating constraints (conditions, properties) which must be satisfied by a solution of the problem. For example, consider a bicycle number lock. We forgot the first digit, but remember some constraints about it: The digit was an odd number, greater than 1, and not a prime number. Combining the pieces of partial information expressed by these constraints (digit, greater than 1, odd, not prime) we are able to derive that the digit we are looking for is “9”.

*Constraints* can be considered as pieces of partial information. Constraints describe properties of unknown objects and relationships between them. Constraints are formalized as distinguished, predefined predicates in first-order predicate logic. The objects are modeled as variables.

Constraints allow for a finite representation and efficient processing of possibly infinite relations. For example, each of the two arithmetic constraints  $X+Y=7$  and  $X-Y=3$  admits infinitely many solutions over the integers. Taken together, these two constraints can be simplified into the solution  $X=5$  and  $Y=2$ .

From the mid-1980's, *constraint logic programming* combined the advantages of *logic programming* and *constraint solving*. Constraint-based programming languages enjoy elegant theoretical properties, conceptual simplicity, and practical success.

In *logic programming languages*, problem-solving knowledge is stated in a declarative way by rules that define relations. A solution is searched for by applying the rules to a given problem. A fixed strategy called *resolution* is used.

In *constraint solving*, efficient special-purpose algorithms are employed to solve sub-problems expressed by constraints.

As it runs, a *constraint program* successively generates constraints. As a special program, the *constraint solver* stores, combines, and simplifies the constraints until a solution is found. The partial solutions can be used to influence the run of the program.

The advantages of constraint logic programming are: declarative problem modeling on a solid mathematical basis, propagation of the effects of decisions using efficient algorithms, and search for optimal solutions.

The use of constraint programming supports the complete software development process. Because of its conceptual simplicity and efficiency *executable specifications*, *rapid prototyping*, and *ease of maintenance* are achievable.

Already since the beginning of the 1990's, constraint-based programming has been commercially successful. In 1996, the world wide revenue generated by constraint technology was estimated to be on the order of 100 million dollars. The technology has proven its merits in a variety of application areas, including decision support systems for scheduling, timetabling, and resource allocation.

For example, the system Daysy performs short-term personnel planning for Lufthansa after disturbances in air traffic (delays, etc.), such that changes in the schedule and costs are minimized. Nokia uses constraints for the automatic configuration of software for mobile phones. The car manufacturer Renault has been employing the technology for short-term production planning since 1995.

## Overview of the Book

This book is intended as a concise and uniform overview of the fundamentals of constraint programming: languages, constraints, algorithms, and applications.

The first part of the book discusses classes of constraint programming languages. The second part introduces types of constraints and algorithms to solve them. Both parts include examples. The third part describes three exemplary applications in some detail. In the appendix, we briefly give syntax and semantics of first-order predicate logic which constitutes the formal basis of this book.

In the first part of the book, we introduce the basic ideas behind the classes of (concurrent) constraint logic programming languages in a uniform abstract framework.

In Chap. 4, we introduce logic programming. We define syntax, operational semantics in a calculus, and declarative semantics in first-order logic. We give soundness and completeness results that explain the formal connection between operational and declarative semantics. With Prolog we briefly introduce the best known representative and classic of logic programming languages.

Step by step we extend this class of programming languages in the following chapters. We will keep the structure of presentation and emphasize the commonalities and explain the differences.

In Chap. 5, we extend logic programming by constraints, leading to constraint logic programming. In Chap. 6, constraints present themselves as a

formalism for communication and synchronization of concurrent processes. In Chap. 7, we introduce a concurrent programming language for writing constraint solvers and constraint programs called Constraint Handling Rules.

In the second part of the book, we explain what a constraint solver does and what it should do. We define the notion of constraint system and explain the principles behind constraint-solving algorithms such as variable elimination and local-consistency techniques. We introduce common constraint systems such as Boolean constraints for circuit design, terms for program analysis, linear polynomial equations for financial applications, finite domains for scheduling, and interval constraints for solving arbitrary arithmetic expressions.

Constraint Handling Rules will come in handy to specify and implement the corresponding constraint-solving algorithms at a high level of abstraction. We will analyze termination, confluence, and worst case time complexity of the algorithms. For each constraint system, we give an example of its application.

In the third part of the book, we reach the commercial practice of constraint programming: We briefly describe the market for this technology, the involved software companies, applications areas, and sample concrete projects. Then we present in more detail three applications: from timetabling to internet-based rent advice and optimal placement of senders for wireless communication.

References to related literature and a detailed index conclude the book.

Since this book concentrates on the essentials of constraint programming and reasoning, it does not address the following topics: temporal and spatial constraints, dynamic (undoable) constraints, soft (prioritized) constraints, constraint-based optimization techniques, low-level implementation techniques, programming methodology, non-logic programming languages (functional, object oriented, imperative) with constraints, and databases with constraints.

A final remark: The web pages of this book at <http://www.informatik.uni-ulm.de/pm/mitarbeiter/fruehwirth/pisa> contain teaching aids like slides and exercises, as well as links to programming languages, tutorials, software, further references, and more.

Essentials of Constraint Programming

Frühwirth, T.; Abdennadher, S.

2003, IX, 147 p., Hardcover

ISBN: 978-3-540-67623-2