Foreword

Coordinating production across a supply chain, designing a new VLSI chip, allocating classrooms or scheduling maintenance crews at an airport are just a few examples of complex (combinatorial) problems that can be modeled as a set of decision variables whose values are subject to a set of constraints. The decision variables may be the time when production of a particular lot will start or the plane that a maintenance crew will be working on at a given time. Constraints may range from the number of students you can fit in a given classroom to the time it takes to transfer a lot from one plant to another. Despite advances in computing power, many forms of these and other combinatorial problems have continued to defy conventional programming approaches.

Constraint Logic Programming (CLP) first emerged in the mid-eighties as a programming technique with the potential of significantly reducing the time it takes to develop practical solutions to many of these problems, by combining the expressiveness of languages such as Prolog with the computational power of constrained search. While the roots of CLP can be traced to Monash University in Australia, it is without any doubt in Europe that this new software technology has gained the most prominence, benefiting, among other things, from sustained funding from both industry and public R&D programs over the past dozen years. These investments have already paid off, resulting in a number of popular commercial solutions as well as the creation of several successful European startups.

This book is about DiSCiPl, a two-and-a-half-year European project aimed at paving the way to broader adoption of CLP. DiSCiPl stems from the observation that, while CLP can significantly reduce the time it takes to develop practical solutions to complex combinatorial problems, doing so often involves a lot of tinkering and deep insight into the innerworkings of the language and its underlying search mechanisms. The objective of the project, which was launched in late 1996 in the context of the European Research Program in Information Technology (ESPRIT), was to research and validate new concepts and tools aimed at significantly facilitating the development and refinement of CLP programs, with a special focus on “Constraint Debugging”. “Debugging” here is to be interpreted in the broad sense and includes both "correctness debugging", namely ensuring that a CLP program properly captures all aspects of a given problem, and "performance debugging", which has to do with analysing and fine-tuning performance of CLP programs.

DiSCiPl brought together some of the best brains in the field in Europe, combining participation of four leading research organisations (INRIA in association with ERCIM, UPM, the University of Linköping and the University of Bristol), two CLP vendors (Cosytec and PrologIA) and two solution providers (ICON and OM Partners). The results, which are presented in this book, include a novel methodology for CLP debugging together with a rich collection of new debugging techniques. At the time of writing, some of these techniques have already found their way into a number of popular CLP packages, making their benefits available to a sizable user population. A nice feature of the book is its discussion of user cases, detailing these benefits. Beyond its more immediate impact, the DiSCiPl project has also produced significant theoretical results that open the door to new and exciting avenues for future research.

It has been a privilege and a pleasure to work with such an enthusiastic group of people from the very inception of their project and to see many of their results so quickly made available to the user community. I hope that you will enjoy reading this book as much as I have.

January 2000

Norman M. Sadeh
European Commission

Preface

This book is the first one entirely dedicated to the problem of **Constraint Debugging**. It presents new approaches to debugging for the computational paradigm of **Constraint Programming** (CP).

Conventional programming techniques are not well suited for solving combinatorial problems in industrial applications like scheduling, decision making, resource allocation, or planning. Constraint Programming offers an original approach allowing for efficient and flexible solving of complex problems, through combined implementation of various constraint solvers, expert heuristics, and programmed search. As an emerging software technology, Constraint Programming is relatively young compared to other languages, technologies, and paradigms. However, it has already produced convincing results and its applications are increasingly fielded in various industries.

One of the remaining shortcomings of CP technology is that it is still somewhat difficult to use. This is due to the intrinsic complexity of the problem areas tackled and to the comparatively sophisticated solutions offered by this technology. These difficulties can be overcome by a combination of adequate training and the availability of effective debugging techniques and environments adapted to the particular characteristics of the paradigm. In fact, one of the main features of CP is a new approach to software production: the same program is progressively improved at each step of the development cycle, from the first prototype until the final product. This makes debugging one of the cornerstones of CP technology.

Debugging is considered here in a broad sense: it concerns both validation aspects (to build a correct application) as well as methodological aspects (to find the best solution to a problem by gaining a better understanding of constraint solver behavior). To satisfy these objectives, tools must locate and explain bugs, and graphical tools must help in the process of interpreting program behaviour and results. The debugging tools of the commercial CP systems are often ineffective in industrial situations, and the debugging techniques originating from imperative languages are mostly inapplicable to CP. The main reason is that the huge numbers of variables and constraints make the computation state difficult to understand, and that the non-deterministic execution drastically increases the number of computation states which must be analysed.

This book contains most of the results of the DiSCiPl project. DiSCiPl (Debugging Systems for Constraint Programming) is a recently completed European (IT4) reactive Long Term Research project which had been running for over two and a half years, from October 1996 to June 1999, with four academic (INRIA-Rocquencourt, manager, with ERCIM, UPM, University of Linköping, University of Bristol) and four industrial partners (Cosytec, PrologIA, ICON, OM Partners). The objectives were to develop the theory of constraint debugging and to create tools to help the programmer during all phases of development.

DiSCiPl has produced a good number of results at both the theory and implementation levels. The new theoretical results in debugging have been cast into the form of a practical "DiSCiPl debugging methodology". These practical developments have produced enhanced versions of industrial and academic Constraint Logic Programming (CLP) platforms (specifically Prolog IV, Chip++5.2.1, GNU-Prolog and Ciao/Prolog) with new, rich debugging capabilities.

This book is a first attempt to give a unified view of "constraint debugging" and it presents the results of the DiSCiPl project in a more comprehensive manner. Technical details can be found in various publications originated from the project or in the public deliverables available at the URL http://discipl.inria.fr.

The DiSCiPl project allowed making significant progress towards understanding the problem of constraint debugging and produced a good number of results and tools. It however did not close the topic. On the contrary we believe that it has opened a field, showing that debugging is an essential part of the process of constraint programming. Only some aspects of debugging have been explored and only still incomplete tools have been produced in the limited duration of the project. They are starting points. Specific suggestions for future work in the area of constraint debugging are presented in different chapters of this volume.

The book consists of an introduction and three parts, each of them composed of several chapters. Most of the chapters can be read independently. The introduction (chapter 1) presents the "DiSCiPl debugging methodology" and explains how all chapters are related.

- Part 1: **Correctness Debugging** Five chapters presenting techniques and tools for finding the reasons for incorrect behaviour of a constraint program. Most of them use assertions and static analysis techniques.
- Part 2: **Performance Debugging** Seven chapters presenting visualization tools which facilitate understanding of the search space and of the constraint propagation. They facilitate finding the reasons for inefficient execution (performance debugging) and they may also contribute to correctness debugging.

– Part 3: **User cases** One chapter which presents feedback from the use of some of the debugging tools in an industrial context.

The research presented in this book has been influenced by the feedback obtained during the project reviews from the the reviewers Dominique Bolignano and Seif Haridi, and the project officer, Norman M. Sadeh. We are very grateful to all of them. The members of the Advisory Committee followed the project giving us very useful and sometimes very enthusiastic feedback.

In addition to European ESPRIT LTR project DiSCiPl # 22532 this work has been partially supported by Spanish CICYT Projects TIC97-1640-CE and TIC99-1151.

April 2000

P. Deransart
M. Hermenegildo
J. Małuszyński