6. Parallel Algorithms for String Pattern Matching, Costas Iliopoulos
7. Design of Parallel Matrix Algorithms, D. J. Evans
8. Resilient Parallel Computing on Unreliable Parallel Machines, Z. M. Kedem, K. V. Palem, A. Raghunathan, and P. G. Spirakis
9. P-Completeness, Jacobo Torán
10. An Introduction to Distributed Memory Models of Computation, Alan Gibbons
11. Network Orientation, Gerard Tel
12. Special Purpose Parallel Computing, W. F. McColl
13. General Purpose Parallel Computing, W. F. McColl
14. Complexity Models for All-Purpose Parallel Computation, Andrew Chin
15. Implementing Sparse BLAS Primitives on Concurrent/Vector Processors, Harry A. G. Wijshoff

IAN PARBERRY
University of North Texas

JIM WALDO (ed.)
The Evolution of C++. MIT Press. 1993. ISBN 0-262-73017-X. £18.95. 279 pp. softbound.

The language C++ started becoming widely available in early 1985 with the first public release of the AT&T compiler cfront, release E. Work had started many years previously, in 1979, developing through Cpre and C with Classes. The name C++ came into being in late 1983 during the preparation of the first edition of *The C++ Reference Manual*. Since cfront release E, the language has grown steadily in capability and popularity, so that today it is used for tasks varying from SLR camera firmware to controlling aircraft landing gear, and by many large software producers. The process of standardizing C++ started in late 1989 with the formation of the ANSI C++ committee, X3J16, and turned into an international process in 1991 when the ISO C++ committee was formed. This process is currently scheduled to finish in late 1996.

During this process C++ has gone through many changes and this book attempts to provide some insight into the reasoning behind these changes. It is a collection of papers from many different people and at different stages in the evolution of the language. These papers give a glimpse of what was going through the minds of influential C++ developers and users, and an indication of why the language has developed in the directions it has. Of course, Bjarne Stroustrup's name appears several times as author or joint author of papers in the book, but other less well known but nevertheless influential people are also represented. Some of them are now actively involved in the standardization process.

The papers, 14 in all, are divided into five groups: The Early Years, Multiple Inheritance, Exception Handling, Runtime Typing and Distributed Computing. The two 'Early Years' papers by Stroustrup together give a good introduction to what guided him in designing the language. The paper by Koenig and Stroustrup on exception handling, and the paper by Lenkov, Mehta and Unni on runtime type information were both used as a basis for the design of the corresponding feature as it is in C++ today. Multiple inheritance is still a subject of much disagreement, and the papers by Cargil and Waldo on multiple inheritance provide a good summary of the arguments for and against this language feature. The remaining papers cover more general topics like type hierarchy and class design, and designs of specific libraries like InterViews and the Arjuna distributed programming system.

Does the book succeed in its aim? Not entirely. The author admits in the introduction that it is very difficult, if not impossible, to collect together a small set of papers that cover all aspects of the development of C++. While all of the papers in this particular collection are interesting in their own right, the connections of some of them to the development of C++ are not always clear. In addition, a collection of individual papers necessarily provides only a disjointed view of history, giving little feel for the active discussions and development work that took place in between. If you want a collection of papers on C++, this is a good place to start. If you want a better understanding of how C++ has developed over the years, and is still developing, try *The Design and Evolution of C++*, by Stroustrup himself.

S. RUMSBY
University of Warwick

FRÉDÉRIC BENHAMOU AND ALAIN COLMERAUER (eds)
Constraint Logic Programming: Selected Research. MIT Press. 1993. ISBN 0 262 02353 9. £44.95. 486 pp. hardbound.

EUGENE C. FREUDER AND ALAN K. MACWORTH (eds)
Constraint-Based Reasoning. MIT Press. 1994. ISBN 0 262 56075 5. £31.50. 403 pp. softbound.

Both of these books are about constraints (as their titles suggest), both are collections of papers, and both are from the same publisher. *Constraint Logic Programming* is a collection of papers from an international workshop, whereas *Constraint-Based Reasoning* is a reprint of a special issue of *Artificial Intelligence Journal* devoted to the general problem of constraints in Artificial Intelligence (AI). From these remarks, it should come as no surprise that the intended audiences of the two collections are quite different: *Constraint Logic Programming* comes from a Prolog background and is intended for the logic programming community, while *Constraint-Based Reasoning* is for a more general audience.

Constraint-based problem solving and constraint logic programming may be concepts unfamiliar to some readers, so I will give a thumbnail sketch of the basic issues. Constraints in problem solving in general are based on the observation that solutions to problems can

be more rapidly found by constraining the solution space. More generally, in AI, constraints are applied to remove parts of the search space that do not have certain properties: these properties are known *a priori* to define the range of possible solutions. Constraint logic programming is based on two observations. The first is that constraints can be expressed as predicates and relations over appropriate universes. The second is that constraint satisfaction can be viewed as a species of search in a similar way to the search processes used in Prolog and other logic programming languages. In logic programming languages, solutions are found via search and this can be extended to constraints (one can, in a sense, describe resolution, as well as other theorem-proving and problem-solving methods, as a form of constraint satisfaction). A significant problem with constraint logic programming is that the search spaces are always very large; special properties of search spaces must be found and exploited in order to render constraint logics more tractable computationally: this is the motivation for much research in constraint logic programming.

I will now consider each book in turn, starting with *Constraint Logic Programming*, and then present some more general thoughts on them.

Constraint Logic Programming is divided into four sections, each dealing with a significant issue in constraint logic programming. The first section, consisting of three chapters, deals with the theoretical issues behind the concept of constraint logics. The second section deals with numerical constraints, perhaps the most obvious form of constraint, and, perhaps the most widely applicable. There are seven chapters in Section 2, one of which (Chapter 9) presents an application to robotics. For a numerical constraint system to be of value, it must be applicable to the reals (e.g. if someone wanted to build an expert system for intensive-care patient monitoring, using constraint logic, real valued constraints would be essential), as well as to the rationals and integers. The seemingly simpler problems of constraints defined over the booleans (two element sets, more generally) are considered in Section 3 (which has eight chapters). Boolean constraints are certainly not a curiosity because they represent a constraint interpretation of propositional logic. In this section, the boolean algorithms used in the Prolog-III programming languages are considered (Prolog-III also contains numeric constraint satisfaction algorithms). The final section of the book is concerned with the design of constraint logic languages and consists of six chapters.

Constraint-Based Reasoning contains 12 chapters, including an introductory one. The topics of the chapters are varied, reflecting the different approaches to and uses of constraints in AI; only one of the chapters (the one by van Hentenryck, Simonis and Dincbas) is devoted to constraint logic programming. The first chapter is an introduction to constraint systems, and starts by explaining the concept in terms of the process of finding values of sets of variables and then moves on to the concept of constraint in terms of first-order logic (constraints have long been considered in terms of logic, but constraint logic programming is a relatively new development). The second chapter is about the observation that a considerable amount of effort can be expended during constraint satisfaction on finding total solutions. This can be wasteful because there may be no way to satisfy all the constraints in a system or there may be many (potentially an infinite number of) solutions, only some of which may be optimal. In some (many?) cases, partial solutions (in the sense of solutions that satisfy only some of the constraints in a system) may be acceptable. The process of finding partial solutions is computationally less expensive. Interval-based approaches are the topic of the fourth chapter: as with the first and fourth chapters, this has clear connections with constraint logic programming. Constraints and scheduling are the subject of the fifth chapter, a chapter dealing with a clear (and important) application of constraints; applications are also the subject of Chapters 8 (identification of structure), 9 (scheduling again, but this time with learning), 10 (temporal information) and 12 (planning). Chapter 11 deals with geometry, another area in which constraints arguably play an important part.

Having considered these two collections, a number of points are worth making. The first is that *Constraint Logic Programming* appears to present a more unified front than does *Constraint-Based Reasoning*. Within the constraint logic paradigm, there is an acceptance of a core set of concepts and methods, there is clearly room for extension, but the basics are in place. This contrasts strongly with the position in *Constraint-Based Reasoning*. Within AI in general, approaches to constraints will appear as diverse as they did 10 years ago. This gives *Constraint-Based Reasoning* a somewhat rag-bag feeling. There are clear reasons for this: general tools and techniques for manipulating are not available in AI, yet a general framework of logic forms the core for constraint logic. The objects manipulated in AI are more diverse than in constraint logic, the control structure differs more considerably, and the demands placed upon a constraint satisfaction module are more varied in their specification than they are for constraint logics. The overall impression I gained from these two books is that *Constraint Logic Programming* is more focused in its approach, whereas *Constraint-Based Reasoning* is more varied.

I would suggest that neither book is for general consumption. Both constraint logics and constraints in general are relatively specialized topics, although constraints have been an important issue in AI since the early days (as noted in the introductory paper to *Constraint-Based Reasoning*, a paper on constraints appeared in the first issue of *Artificial Intelligence Journal*). Both books will be of interest to some AI and Cognitive Science researchers, but those with

subscriptions or access to *Artificial Intelligence Journal* will be able to read *Constraint-Based Reasoning* without purchasing the book. *Constraint Logic Programming* is, I tend to think, somewhat frighteningly formal for most readers (although logicist AI workers may well find it of interest). A significant problem with *Constraint Logic Programming* is that it does not contain enough examples of applications: some more application case studies would be welcome. Indeed, examples of the applicability and generality of the approach would be of considerable use in making the techniques more widely known (there are already some parsers for English and other languages that have been constructed, and other applications come immediately to mind) and would assist their argument that constraint logic programming is superior to the better-known form of logic programming. In the area of application and diversity of approach, constraint-based processing in AI and *Constraint-Based Reasoning*, in particular, clearly scores over *Constraint Logic Programming*.

One significant point worth mentioning is the price of these books. Both books are very highly priced. The cost of a softbound book (*Constraint-Based Reasoning*) does seem very high (especially when one considers that the material is reprinted). Although *Constraint Logic Programming* is hardbound, it contains research papers that *could* become out-dated within a relatively short time. Certainly, the quality in terms of production of the latter book is very high, but it, like *Constraint-Based Reasoning*, would probably be better bought by libraries than by individuals because of their cost.

Despite some negative remarks, I must state that I found both of these books interesting.

IAIN CRAIG
University of Warwick