TLP **4** (1 & 2): 233–237, 2004. © 2004 Cambridge University Press
DOI: 10.1017/S1471068403001935 Printed in the United Kingdom

Book reviews

Programming Constraint Services: *High level Programming of Standard and New Constraint Services* by Christian Schulte, published in 2002 by Springer in the series *Lecture Notes in Artificial Intelligence*, vol. 2302, ISBN 3-540-43371-6, xii + 176 pages, paperback.

'Service' is a word of fashion, from web services to service-oriented software architectures. Schulte brings it to Constraint Programming, crafting a new name, "Constraint Services" for generic methods related to the implementation of search in CP systems. This book, published last year in the LNAI series reviews the theory and implementation of search in the Oz/Mozart concurrent programming language.

The book is intended for all CP practitioners, from newcomers wanting to learn how a search exploration is built to experts looking for ideas in designing, or improving their favourite CP system.

I enjoyed reading the book, found it enlightening and, at times, frustrating. Talk about concurrent feelings?

What's in the book

It is definitely an enlightening reading.

To my knowledge, it is the first book that thoroughly goes into the implementation issues of a CP system. As such, it does fill an important hole in the shelves of our libraries. It is a pity that such topics have been neglected and I am grateful to Christian Schulte for having the talent and taking the time to explain how a constraint system is built, both from theoretical and practical points of view.

The book indeed provides a self-contained and very didactic presentation of the Oz/Mozart platform. The theory of the concurrent language is based on the introduction of computation spaces which are entities that model the context of computation, such as the state of domains. The computation spaces are explicit: they are available as "first class citizens" of the language, as stated by the author. As such, they can be programmed, by being passed as functional arguments. This framework supports, among other things, a very simple way of programming search by copy.

The book features 15 chapters. Chapters 1,2,3 provide introductions, including a self contained description of Oz-light, a narrowed down version of the Oz system. Chapters 4 to 7 present the theory of global search in a concurrent language environment, including the issue of copying vs. re-computation Chapters 8 and 9 present two applications of this theory, to interactive search (supported by the 'Oz explorer' tool) and to distributed search. Chapters 10 and 11 develop the full theoretical model of search combinators, which are the objects that handle

https://doi.org/10.1017/S1471068403221931 Published online by Cambridge University Press

the computation spaces to control the search. Chapters 12 and 13 focus on implementation issues while the chapters 14 and 15 list other approaches and conclude.

The theoretical apparatus developed by Christian Schulte is very elegant and the author goes into many subtleties of tree search. Among them, here are a few original topics discussed or defended by the author that particularly caught my interest:

- It is claimed that the structure of the search tree (deciding what branching decisions structure the set of states that are built) and its exploration strategy (deciding the order in which the states are opened) are orthogonal concerns and should therefore be handled independently,
- ways of turning n-ary splitting schemes into binary alternatives are discussed,
- new ways of combining re-computation and copy are proposed. Although the issue is complex, the author manages to make it accessible, thanks to the small didactic of Oz light in chapter 3. This particular chapter is very useful: as a frequent reader of Christian Schulte's papers on the subject, I happily confess that the book, thanks to the presentation effort, is much more accessible than earlier papers.

In one sentence rather than a thousand words, the book provides an easy access to a versatile technology and a complete CP system. Congratulations for making things look simple and natural.

What's not in the book

Now, on the frustrating side, the weakness of the book is the lack of an account of related work. Although a small chapter lists references to other systems, the core presentation is focused on the sole Oz/Mozart platform. The interested reader would like to understand, which of the techniques mandatorily require working within a concurrent programming language and which do not. Unfortunately, such insights are not provided by the author. This is a pity, in particular concerning the section devoted to search by re-computation.

An approach using search by re-computation has been proposed by Laurent Perron and marketed by Ilog in the "Parallel Solver" product, and it would be most useful for the reader to understand how the approaches differ and what common procedures they use. In my opinion, re-computation-based search strategies do not require a concurrent programming environment, as they can manipulate a data structure representing the tree of explored nodes in a standard object-oriented environment, without sacrificing either the elegance of the code or the performance of the exploration. As a reader, I would be interested in either understanding that my intuition is wrong and why, or that it may be right under some conditions and that some of the services from the Oz system can be described as primitives in a non-concurrent environment, and related to mechanisms of Perron's architecture.

https://doi.org/10.1017/S1471068403221931 Published online by Cambridge University Press

So, should the book be recommended?

In conclusion, the book is a wonderful presentation of the research agenda conducted by Christian Schulte and of the achievements he and his team have realized in the past few years around the Oz/Mozart system. As such, the book is of prime interest to designers of CP systems as well as researchers who are curious about the virtues of concurrent programming languages.

The presentation of "Constraint Services" may not yet be the review of the state of the art techniques on search for constraint programming. To date, such a book still does not exist. But who knows, reading some of the future work directions, it could well be Christian's next *grand oeuvre*.

François Laburthe
Bouygues, France

Knowledge Representation, Reasoning and Declarative Problem Solving by Chitta Baral, Cambridge University Press, 2003. ISBN 0-521-81802-8 (hardback), xiv + 530 pages.

The book provides a comprehensive overview of several logic programming formalisms based on the language of normal logic programs and its extensions that allow disjunctions in the head and both strong and default negations. The most broadly accepted semantics for programs in such general syntax is the semantics of answer sets, which generalizes the stable-model semantics of normal logic programs. Recent research demonstrated that logic programs, when interpreted by the answer-set semantics, can be used in declarative programming in a way that abandons some major tenets of the standard logic programming as implemented and exemplified by Prolog. Namely, to solve a problem the programmer constructs a program so that problem solutions are represented by answer sets of the program, and not by variable substitutions. Computing models and answer sets, rather than resolution refutation proofs, becomes the fundamental computational task and a uniform control mechanism.

The book refers to logic programming formalisms that are so used by a single term AnsProlog (with some modifiers specifying the exact set of connectives that are allowed). It stands for **Pro**gramming in **log**ic with **Ans**wer sets and is meant to stress that one programs by writing theories in the language of logic and to emphasize a shift in focus from proofs and variable substitutions to answer sets (models).

In the text, the author introduces AnsProlog, presents basic theoretical results and addresses the issue of programming methodology. A substantial fragment of the book deals with applications of AnsProlog in solving combinatorial and constraint satisfaction problems, and in knowledge representation. In the latter case, the author focuses on applications to reasoning about action and planning.

The book consists of nine chapters and two appendices:

1. Declarative programming in AnsProlog*: introduction and preliminaries.
2. Simple modules for declarative programming with answer sets.

https://doi.org/10.1017/S1471068403221931 Published online by Cambridge University Press

3. Principles and properties of declarative programming with answer sets.
4. Declarative problem solving and reasoning in AnsProlog*.
5. Reasoning about actions and planning in AnsProlog*.
6. Complexity, expressiveness, and other properties of AnsProlog* programs.
7. Answer set computing algorithms.
8. Query answering and answer set computing systems.
9. Further extensions of and alternatives to AnsProlog*.

Appendix A: Ordinals, lattices, and fixpoint theory
Appendix B: Turing machines

Chapter 1 contrasts declarative and procedural programming, and points to the importance of the former for applications in artificial intelligence. It provides an introduction to the syntax and the semantics of AnsProlog. Chapter 2 shows several small programs solving problems and subproblems frequently found in combinatorial and constraint satisfaction problems (generating enumerations and orderings, imposing cardinality and weight constraints, checking satisfiability of propositional formulas) and in knowledge representation (modeling normative statements, the frame problem). Chapter 3 discusses basic theory of AnsProlog presented from the perspective of systematic program development. Chapter 4 discusses applications of AnsProlog in solving combinatorial and constraint satisfaction problems. It also shows AnsProlog representations of prioritized default theories and inheritance hierarchies. Chapter 5 deals with reasoning about action and planning and develops AnsProlog representations for several versions of problems in these domains. The complexity of reasoning with several classes of AnsProlog programs and the expressive power of the corresponding languages is the subject of Chapter 6. This chapter also deals with the issues of compactness of representation and compilability, and relates AnsProlog to other knowledge representation formalism such as circumscription, autoepistemic logic and default logic. Chapter 7 presents algorithms to compute answer sets, as well as the well-founded semantics of AnsProlog programs (the latter is important as it approximates the answer-set semantics and is fast to compute). Chapter 8 presents most advanced implementations of AnsProlog: *smodels* and *dlv*. The last chapter deals with extensions of AnsProlog (for instance, allowing nested expressions) as well as with additional applications. The appendices provide material needed to make the book self-contained.

This book is an important milestone in the development of logic programming as a knowledge representation tool. It is the first book that treats seriously a possibility of using logic programming with answer-set semantics as a declarative programming tool and a knowledge representation formalism and does it in a systematic and comprehensive way. It presents all major results concerning that formalism. It comes with an impressive number of examples and example programs. Especially important from the perspective of applications and ultimate acceptance of the paradigm of answer-set programming is an elegant treatment of reasoning about action and planning problems.

The book is a very good reference text. Sections containing notes and comments provide good insights into the time line of the development of ideas, and provide

https://doi.org/10.1017/S1471068403221931 Published online by Cambridge University Press

pointers to all relevant bibliography. It will be greatly appreciated by the researchers in the field.

This text is an excellent research monograph that will certainly stimulate and affect further research efforts in the area of answer-set programming. It is also a text that could be used in the classroom for teaching declarative problem solving and knowledge representation.

The book is not free of shortcomings, which is perhaps not surprising given the novelty of the subject and the size of the text. I felt that the organization of the material might be improved. For instance, it is not clear why the material on default logic and inheritance networks was included in Chapter 4 and, similarly, why connections of AnsProlog with circumscription, default logic and autoepistemic logic were dealt with in Chapter 6. The well-founded semantics is referred to in Chapter 7 and, in fact, is needed there to put some of the algorithms in the proper context, yet its discussion is delayed to Chapter 9.

As mentioned earlier, the text contains numerous example programs. While intuitively these programs are correct, I missed formal correctness proofs. They could easily have been provided, at least for some of the programs (for instance, those presented in Chapter 4). It is even more important if one remembers that one of the selling points of declarative programming is that it facilitates program correctness verification.

The book does an excellent job demonstrating that AnsProlog is a powerful modeling language. Relatively little attention was devoted to matters of computational effectiveness. I would welcome comments indicating the size of problems that can be handled by current generation of AnsProlog implementations. In this context, I felt the book did not give enough justice to extensions of AnsProlog allowing explicit modeling of cardinality and weight constraints. After all, it is well known that they are critical for the success of the current *smodels* implementation and without them, the performance suffers.

Finally, the title of the book is perhaps somewhat misleading. It suggests a general text on declarative programming and knowledge representation, while the book is really about AnsProlog and its applications.

However, all these, overall minor, shortcomings take second place to what is really important here. The area of declarative programming with the focus on models and not proofs has its first monograph. The presentation is formal yet the text reads well. It is comprehensive and provides an exhaustive collection of references and examples. It also emphasizes the importance of programming methodology and builds its rudiments. I wholeheartedly recommend this book to researchers and students in the fields of logic programming, declarative programming and knowledge representation.

Mirosław Truszczyński
University of Kentucky, Lexington, USA

https://doi.org/10.1017/S1471068403221931 Published online by Cambridge University Press

https://doi.org/10.1017/S1471068403221931 Published online by Cambridge University Press