

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

2962

Springer

Berlin

Heidelberg

New York

Hong Kong

London

Milan

Paris

Tokyo

Stefano Bistarelli

Semirings for Soft Constraint Solving and Programming



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Author

Stefano Bistarelli
Università degli Studi "G. D'Annunzio" di Chieti-Pescara
Dipartimento di Scienze
Viale Pindaro, 42, 65127 Pescara, Italy
E-mail: bista@sci.unich.it
and
Istituto di Informatica e Telematica, C.N.R.
Via G. Moruzzi, 1, 56124 Pisa, Italy
E-mail: stefano.bistarelli@iit.cnr.it

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data is available in the Internet at <<http://dnb.ddb.de>>.

CR Subject Classification (1998): D.3.3, D.1.6, D.3.1, D.3, F.3.1, D.2.4, I.2.2-3

ISSN 0302-9743

ISBN 3-540-21181-0 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10988022 06/3142 5 4 3 2 1 0

Foreword

Constraint satisfaction and constraint programming have shown themselves to be very simple but powerful ideas. A constraint is just a restriction on the allowed combinations of values for a set of variables. If we can state our problem in terms of a set of constraints, and have a way to satisfy such constraints, then we have a solution. The idea is general because it can be applied to several classes of constraints, and to several solving algorithms. Moreover, it is powerful because of its unifying nature, its generality, its declarative aspects and its application possibilities. In fact, many research and application areas have taken advantage of constraints to generalize and improve their results and application scenarios.

In the last 10 years, however, this simple notion of constraint has shown some deficiencies concerning both theory and practice, typically in the way over-constrained problems and preferences are treated. When a problem has no solution, classical constraint satisfaction does not help. Also, classical constraints are not able to model conveniently problems which have preferences, for example over the selection of the most relevant constraints, or about the choice of the best among several solutions which satisfy all the constraints.

Not being able to handle non-crisp constraints is not just a theoretical problem, but it is also particularly negative for applications. In fact, over-constrained and preference-based problems are present in many application areas. Without formal techniques to handle them, it is much more difficult to define a procedure which can easily be repeated to single out an acceptable solution, and sometimes it is not even possible.

For this reason, many researchers in constraint programming have proposed and studied several extensions of the classical concepts in order to address these needs. This has led to the notion of *soft constraints*. After several efforts to define specific classes of soft constraints, like fuzzy, partial and hierarchical, the need for a general treatment of soft constraints became evident, a treatment that could model many different classes altogether and to prove properties for all of them. Two of the main general frameworks for soft constraints were *semiring-based soft constraints* and *valued constraints*.

This book is a revised, extended version of the Ph.D. thesis of Stefano Bistarelli, whom we had the pleasure to supervise at the University of Pisa. It focuses mainly on the semiring-based soft constraint formalism, also comparing it with many of the specific classes and also with valued constraints. Semiring-based soft constraints are so called because they are based on an un-

derlying semiring structure, which defines the set of preferences, the way they are ordered, and how they can be combined. This concept is very general and can be instantiated to obtain many of the classes of soft constraints that have already been proposed, including their solution algorithms, and also some new ones.

The book includes formal definitions and properties of semiring-based soft constraints, as well as their use within constraint logic programming and concurrent constraint programming. Moreover, it shows how to adapt to soft constraints some existing notions and techniques, such as abstraction and interchangeability, and it shows how soft constraints can be used in some application areas, such as security.

This book is a great starting point for anyone interested in understanding the basics of semiring-based soft constraints, including the notion of soft constraint propagation, and also in getting a hint about the applicability potential of soft constraints. In fact, it is the first book that summarizes most of the work on semiring-based soft constraints. Although most of its content also appears in published papers, this is the only place where this material is gathered in a coherent way.

This book is the result of several threads of collaborative work, as can be seen from the many publications that are cited in the bibliography and whose content is reflected in the book. Therefore many authors have contributed to the material presented here. However, Stefano Bistarelli succeeded in providing a single line of discourse, as well as a unifying theme that can be found in all the chapters. This uniform approach makes the material of this book easily readable and useful for both novice and experienced researchers, who can follow the various chapters and find both informal descriptions and technical parts, as well as application scenarios.

November 2003

Ugo Montanari¹ and Francesca Rossi²

¹ Dipartimento di Informatica
Universita' di Pisa
Italy

² Dipartimento di Matematica Pura ed Applicata
Universita' di Padova
Italy

Preface

The *Soft Constraints* idea is able to capture many real-life situations that cannot be represented and solved with the classical crisp constraint framework. In this book we first describe a general framework for representing many soft constraint systems, and then we investigate the related theoretic and application-oriented issues.

Our framework is based on a semiring structure, where the carrier of the semiring specifies the values to be associated with each tuple of values of the variable domain, and the two semiring operations, $+$ and \times , model constraint projection and combination, respectively. The semiring carrier and operations can be instantiated in order to capture all the *non-crisp* constraints representing fuzziness, optimization, probability, hierarchies, and others. The solution of each instance of the soft Constraint Satisfaction Problem (CSP) is computed by using the appropriate \times and $+$ semiring operation.

This uniform representation can be used to give sufficient conditions for the correctness and applicability of local consistency and dynamic programming algorithms. In particular:

- We show that using an idempotent \times operator the classical local consistency (and also dynamic programming) techniques can be used to reduce the complexity of the problem without modifying its solution.
- We adapt to the soft framework partial local consistency and labeling techniques, which require fewer pruning steps of the domain. This means that, although they are able to remove fewer non-optimal solutions than classical algorithms can, partial local consistency algorithms can be beneficial because they are faster and easier implemented.
- We extend general local consistency algorithms that use several pruning rules until the fix-point is reached.

Solving a soft CSP is generally harder than solving the corresponding crisp CSP. For this reason we introduce an abstraction/concretization mapping over soft CSPs in order to solve a problem in an easier environment and then use the abstract results to speed up the search of solutions in the concrete one. Several mappings between existing soft frameworks are given. These mappings will especially be useful for applying soft local consistency techniques in a safe, easy, and faster way. Also useful, when looking for optimal solutions, are the notions of *substitutability* and *interchangeability*. In crisp CSPs they have been used as a basis for search heuristics, solution adaptation, and abstraction techniques.

The next part of the book involves some programming features: as classical constraint solving can be embedded into Constraint Logic Programming (CLP)

systems, so too can our more general notion of constraint solving be handled within a logic language, thus giving rise to new instances of the CLP scheme. This not only gives new meanings to constraint solving in CLP, but it also allows one to treat in a uniform way optimization problem solving within CLP, without the need to resort to ad hoc methods. In fact, we show that it is possible to generalize the semantics of CLP programs to consider the chosen semiring and thus solve problems according to the semiring operations. This is done by associating each ground atom with an element of the semiring and by using the two semiring operations to combine goals. This allows us to perform in the same language both constraint solving and optimization. We then provide this class of languages with three equivalent semantics, model-theoretic, fix-point, and proof-theoretic, in the style of CLP programs. The language is then used to show how the soft CLP semantics can solve shortest-path problems. In a way similar to the soft CLP language we also extend the semantics of the Concurrent Constraints (cc) language. The extended cc language uses soft constraints to prune and direct the search for a solution.

The last part of the book aims to describe how soft constraints can be used to solve some security-related problems. In the framework, the crucial goals of *confidentiality* and *authentication* can be achieved with different levels of security. In fact, different messages can enjoy different levels of confidentiality, or a principal can achieve different levels of authentication with different principals.

Acknowledgement. This monograph is a revised and extended version of my doctoral dissertation which was submitted to the University of Pisa Computer Science Department and accepted in March 2001.

The results presented here have been influenced by many people and I would like to take this opportunity to thank them all.

I wish to thank first of all the supervisor of my Ph.D. thesis Ugo Montanari; the main part of this book came from the research performed under his significant guidance.

I want also to thank Francesca Rossi, my unofficial supervisor; she shared with me the passion for constraints. Many of the ideas collected in this book came from ideas we developed together.

A special thanks is also due to all the friends who shared with me some of the results I collected in this book: Giampaolo Bella, Boi Faltings, Rosella Gennari, H el ene Fargier, Yan Georget, Nicoleta Neagu, Elvinia Riccobene, Thomas Schiex, and G erard Verfaillie.

Many thanks are due to the external reviewers of my Ph.D. thesis, Philippe Codognet and Pascal Van Hentenryck; they carefully read a preliminary version of the thesis and provided many useful comments and suggestions.

My warmest thanks go to my friend Olga Petosa; she was so kind to read all the conversational parts of my work, correct some typing errors, and make it just a little nicer!

Finally, I am also grateful both to the anonymous referees of this book and to Jan van Leeuwen; they gave me a number of hints on a draft version of the book which helped to improve the final presentation.

Some thanks also to Anna Kramer and Alfred Hofmann who helped me with the last details needed for the publication of this book.

Contents

1. Introduction	1
1.1 From the Beginning	1
1.2 Applications	2
1.2.1 Assignment Problems	2
1.2.2 Personnel Assignment	2
1.2.3 Network Management	3
1.2.4 Scheduling Problems	3
1.2.5 Transport Problems	4
1.3 Crisp Constraints	4
1.3.1 CSPs	5
1.3.2 Constraint Logic Programming	12
1.4 Non-crisp Constraints	14
1.4.1 Partial Constraint Satisfaction Problems	15
1.4.2 Constraint Hierarchies	15
1.4.3 Fuzzy, Probabilistic and Valued CSPs	16
1.5 Overview of the Book	16
1.5.1 Structure of Subsequent Chapter	19
1.5.2 The Origin of the Chapters	20
2. Soft Constraint Satisfaction Problems	21
2.1 C-semirings and Their Properties	22
2.2 Constraint Systems and Problems	27
2.3 Instances of the Framework	33
2.3.1 Classical CSPs	33
2.3.2 Fuzzy CSPs (FCSPs)	35
2.3.3 Probabilistic CSPs (Prob-CSPs)	37
2.3.4 Weighted CSPs	38
2.3.5 Egalitarianism and Utilitarianism	39
2.3.6 Set-Based CSPs	40
2.3.7 Valued Constraint Problems	41
2.3.8 N-dimensional C-semirings	49
2.4 Conclusions	50

3. Towards SCSPs Solutions	51
3.1 Soft Local Consistency	52
3.2 Applying Local Consistency to the Instances	59
3.2.1 Classical CSPs	59
3.2.2 Fuzzy CSPs	59
3.2.3 Probabilistic CSPs	60
3.2.4 Weighted CSPs	60
3.2.5 Egalitarianism and Utilitarianism	60
3.2.6 Set-Based SCSPs	61
3.3 About Arc-Consistency in SCSPs	61
3.3.1 Classical Arc-Consistency vs. Semiring-Based One	62
3.3.2 Removing Hidden Variables in SCSPs	65
3.3.3 Tree-Shaped SCSPs	67
3.3.4 Cycle-Cutsets in SCSPs	69
3.4 Labeling and Partial Local Consistency for SCSPs	70
3.4.1 Labeling in SCSPs	72
3.4.2 Partial Local Consistency	76
3.4.3 Domains	79
3.5 Constraint Propagation: Generalization and Termination Condi- tions	82
3.5.1 Some Useful Orderings over Semiring Constraints	83
3.5.2 Order-Related Properties of Soft Local Consistency Rules	86
3.5.3 The Generic Iteration Algorithm	86
3.5.4 Generalized Local Consistency for SCSPs via Algorithm GI	87
3.5.5 Termination of the GI Algorithm over Soft Constraints...	89
3.6 Dynamic Programming for SCSPs	92
3.7 Conclusions	97
4. SCSP Abstraction	99
4.1 Abstraction	101
4.2 Abstracting Soft CSPs	103
4.3 Properties and Advantages of the Abstraction	105
4.3.1 Relating a Soft Constraint Problem and Its Abstract Version	105
4.3.2 Working on the Abstract Problem	111
4.4 Some Abstraction Mappings	117
4.5 Abstraction vs. Local Consistency	120
4.6 Related Work	122
4.7 Conclusions	124
5. Higher Order Semiring-Based Constraints	125
5.1 Domains and Semirings	126
5.2 Constraint Problems and Solutions	127
5.3 A Small Language to Program with Soft Constraints	130
5.4 Solving SCSPs	133
5.4.1 Dynamic Programming Techniques	133

5.4.2	Local Consistency Techniques	134
5.4.3	Extending Local Propagation Rules	134
5.5	Constraint Problem as Semiring	135
5.6	Conclusions	136
6.	Soft CLP	137
6.1	Syntax of SCLP Programs	139
6.2	Model-Theoretic Semantics	142
6.3	Fix-Point Semantics	146
6.4	Proof-Theoretic Semantics	149
6.4.1	Universal Closure	151
6.4.2	Existential Closure	156
6.5	A Semi-decidability Result	157
6.6	SCLPs with no Functions	160
6.7	An Operational Model for the SCLP Language Using ASM	162
6.7.1	The Gurevich's Abstract State Machine	163
6.7.2	The Abstract Operational Model of SCLP	164
6.8	Related Work	168
6.9	Conclusions	169
7.	SCLP and Generalized Shortest Path Problems	171
7.1	Classical SP Problems	172
7.2	Partially-Ordered SP Problems	175
7.3	Modality-Based SP Problems	180
7.4	An SP Algorithm for a Class of SCLP Programs	182
7.5	Best-Tree Search in AND-OR Graphs	183
7.5.1	AND-OR Graphs and Best Solution Trees	184
7.5.2	AND-OR Graphs Using SCLP	186
7.6	Conclusions	189
8.	Soft Concurrent Constraint Programming	191
8.1	Concurrent Constraint Programming	192
8.2	Concurrent Constraint Programming over Soft Constraints	195
8.3	Soft Concurrent Constraint Programming	199
8.4	A Simple Example	202
8.5	Observables and Cuts	203
8.5.1	Capturing Success Computations	204
8.5.2	Failure	207
8.6	An Example from the Network Scenario	208
8.7	Conclusions	212
9.	Interchangeability in Soft CSPs	213
9.1	Interchangeability	214
9.2	Interchangeability in Soft CSPs	216
9.2.1	Degradations and Thresholds	218
9.2.2	Properties of Degradations and Thresholds	221

9.2.3	Computing $\delta/\alpha/\alpha\text{-set}$ -Substitutability/Interchangeability	225
9.3	An Example	229
9.4	Partial Interchangeability	233
9.5	Conclusions	235
10.	SCSPs for Modelling Attacks to Security Protocols	237
10.1	Constraint Programming for Protocol Analysis	240
10.1.1	The Security Semiring	241
10.1.2	The Network Constraint System	242
10.1.3	Computing the Security Levels by Entailment	243
10.1.4	The Initial SCSP	244
10.1.5	The Policy SCSP	245
10.1.6	Assessing the Expected Risk	246
10.1.7	The Imputable SCSPs	248
10.1.8	Formalising Confidentiality	248
10.1.9	Formalising Authentication	250
10.2	An Empirical Analysis of Needham-Schroeder	252
10.3	The Kerberos Protocol	255
10.4	Analysing Kerberos	257
10.4.1	Confidentiality	257
10.4.2	Authentication	260
10.5	Conclusions	260
11.	Conclusions and Directions for Future Work	263
11.1	Summary and Main Results	263
11.2	Directions for Future Research	265
11.2.1	Abstraction	265
11.2.2	High Order Semirings	265
11.2.3	SCLP	265
11.2.4	SCLP for Operational Research Problems	266
11.2.5	Soft Concurrent Constraints	266
11.2.6	Soft Constraints for Security	266
11.2.7	Soft Constraint Databases	267
11.2.8	Soft Web Query	267
References		269

List of Figures

1.1	A possible solution to the 8-queens problem	6
1.2	A CSP which is not solved	9
1.3	Tuple redundancy	12
1.4	The CLP framework.....	13
2.1	Two SCSPs	33
2.2	A CSP and the corresponding SCSP	35
2.3	Combination and projection in classical CSPs.....	36
2.4	From SCSPs to VCSPs	43
2.5	From VCSP to SCSP	45
2.6	From SCSP to VCSP and back to SCSP again.....	46
2.7	From VCSP to SCSP, and to VCSP again.....	47
2.8	Diagrams representing the thesis of Theorem 2.3.11 and 2.3.11 ...	49
3.1	A CSP which is not AC	63
3.2	A fuzzy CSP	63
3.3	An SAC-consistent FCSP and the corresponding AC-consistent CSP	64
3.4	Redundant hidden variables in CSPs.....	66
3.5	Redundant hidden variables in SCSPs	66
3.6	New semiring value for the extended tuple.....	67
3.7	A tree-shaped SCSP and a top-down instantiation order	68
3.8	An SCSP with a cycle over x, y, z and a tree shape over the other variables	69
3.9	The search tree for the SCSP of Figure 3.8	70
3.10	Disjunction example	72
3.11	AC and labeling	73
3.12	A fuzzy CSP and its solution	74
3.13	The set of the $\{x, y\}$ -labelings corresponding to the problem.....	74
3.14	A 3-up-down-stair	81
4.1	A Galois insertion	103
4.2	A fuzzy CSP and its solutions	104
4.3	The concrete and the abstract problem	106
4.4	An example of the abstraction fuzzy-classical	107
4.5	An abstraction which is not order-preserving	108

4.6	The general abstraction scheme, with \times idempotent	112
4.7	The scheme when \tilde{f} does not modify anything	113
4.8	The scheme when the concrete semiring has a total order	114
4.9	An example with \times idempotent	114
4.10	The scheme when \times is not idempotent	115
4.11	An example with \times not idempotent	116
4.12	Several semirings and abstractions between them	119
5.1	A 3-variable fuzzy CSP	127
5.2	The fuzzy CSP after the application of rule r_1	134
7.1	An SP problem	173
7.2	An SP problem with labeled arcs	174
7.3	A multi-criteria SP problem	176
7.4	An SP problem with modalities	181
7.5	An example of an AND-OR graph	184
7.6	An example of an AND tree	185
7.7	A weighted AND-OR graph problem	187
7.8	A typical connector	187
7.9	The best tree corresponding to the program in Table 7.5.2	188
8.1	The SCSP describing part of a process network	210
8.2	The ordered process network	211
9.1	An example of CSP with interchangeable values	214
9.2	An example of CSP with computation of neighborhood inter- changeable values	216
9.3	A fuzzy CSP	218
9.4	Example of a CSP modeling car configuration with 4 variables . . .	230
9.5	Example of how δ -substitutability and α -substitutability varies in the weighted CSP over the values of variable E from Fig. 9.4	230
9.6	Example of how α -set-substitutability varies in the weighted CSP over the values of variable E from Fig. 9.4	231
9.7	Example of how δ -substitutability and α -substitutability varies in the opposite-fuzzy CSP over the values of variable E from Fig. 9.4	232
9.8	Example of how α -set-substitutability varies in the opposite- fuzzy CSP over the values of variable E from Fig. 9.4	232
9.9	Example of a search of α -interchangeability computing by the use of discrimination trees	233
10.1	Computation rules for security levels	243
10.2	Algorithm to construct the policy SCSP for a protocol \mathcal{P}	246
10.3	Implementation for a simple risk function	247
10.4	Algorithm to construct an imputable SCSP for \mathcal{P} (fragment)	249

10.5	The asymmetric Needham-Schroeder protocol	252
10.6	Lowe's attack to the Needham-Schroeder Protocol	253
10.7	Fragment of the initial SCSP for Needham-Schroeder protocol . . .	253
10.8	Fragment of the policy SCSP for the Needham-Schroeder protoc.	254
10.9	Fragment of the Imputable SCSP corresponding to Lowe's attack	255
10.10	The Kerberos layout	256
10.11	The Kerberos protocol	256
10.12	The initial SCSP for Kerberos (fragment)	257
10.13	The policy SCSP for Kerberos (fragment)	258
10.14	An imputable SCSP for Kerberos (fragment)	259

List of Tables

5.1	Relationships between semiring elements.	126
6.1	BNF for the SCLP syntax.	139
6.2	Soft n -queens problem.	140
6.3	Clauses for the 5-queens problem.	141
6.4	Our running example.	144
6.5	The T_P operator applied at the running example.	148
7.1	The SCLP program representing the SP problem in Figure 7.1.	173
7.2	The SCLP program representing the multi-criteria SP problem in Figure 7.3.	177
7.3	The SCLP program representing the SP problem with modalities. . .	181
7.4	The general form of an SCLP program representing an SP problem. .	182
7.5	The system of equations corresponding to the SCLP program form of Table 7.4.	183
7.6	The SCLP program representing the AND-OR graph problem in Figure 7.7.	188
8.1	cc syntax	194
8.2	scc syntax	199
8.3	Transition rules for scc	201
8.4	Failure in the scc language	207