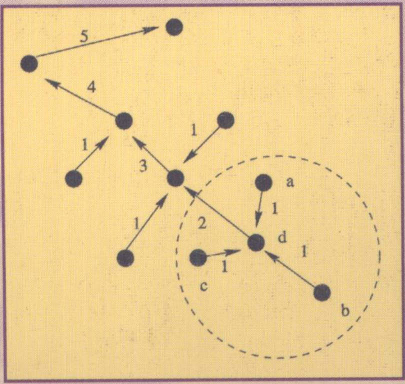
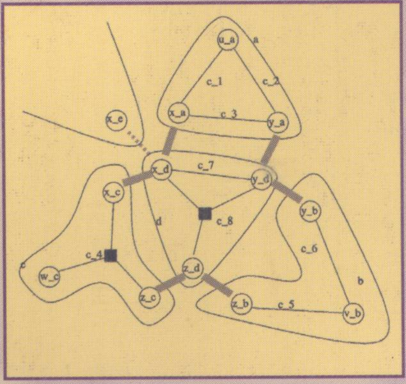
Stefano Bistarelli

LNCS 2962

Semirings for Soft Constraint Solving and Programming





Springer

Stefano Bistarelli

Semirings for Soft Constraint Solving and Programming

Springer

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

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

springeronline.com

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

Lecture Notes in Computer Science 2962

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

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

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 underlying 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 abut 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 Montanari[1] and Francesca Rossi[2]

[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, Hélène Fargier, Yan Georget, Nicoleta Neagu, Elvinia Riccobene, Thomas Schiex, and Gérard 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.

December 2003 Stefano Bistarelli

Lecture Notes in Computer Science

For information about Vols. 1–2840

please contact your bookseller or Springer-Verlag

Vol. 2895: A. Ohori (Ed.), Programming Languages and Systems. XIII, 427 pages. 2003.

Vol. 2894: C.S. Laih (Ed.), Advances in Cryptology - ASIACRYPT 2003. XIII, 543 pages. 2003.

Vol. 2893: J.-B. Stefani, I. Demeure, D. Hagimont (Eds.), Distributed Applications and Interoperable Systems. XIII, 311 pages. 2003.

Vol. 2892: F. Dau, The Logic System of Concept Graphs with Negation. XI, 213 pages. 2003. (Subseries LNAI).

Vol. 2891: J. Lee, M. Barley (Eds.), Intelligent Agents and Multi-Agent Systems. X, 215 pages. 2003. (Subseries LNAI).

Vol. 2890: M. Broy, A.V. Zamulin (Eds.), Perspectives of System Informatics. XV, 572 pages. 2003.

Vol. 2889: R. Meersman, Z. Tari (Eds.), On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops. XIX, 1071 pages. 2003.

Vol. 2888: R. Meersman, Z. Tari, D.C. Schmidt (Eds.), On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. XXI, 1546 pages. 2003.

Vol. 2887: T. Johansson (Ed.), Fast Software Encryption. IX, 397 pages. 2003.

Vol. 2886: I. Nyström, G. Sanniti di Baja, S. Svensson (Eds.), Discrete Geometry for Computer Imagery. XII, 556 pages. 2003.

Vol. 2885: J.S. Dong, J. Woodcock (Eds.), Formal Methods and Software Engineering. XI, 683 pages. 2003.

Vol. 2884: E. Najm, U. Nestmann, P. Stevens (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 293 pages. 2003.

Vol. 2883: J. Schaeffer, M. Müller, Y. Björnsson (Eds.), Computers and Games. XI, 431 pages. 2003.

Vol. 2882: D. Veit, Matchmaking in Electronic Markets. XV, 180 pages. 2003. (Subseries LNAI).

Vol. 2881: E. Horlait, T. Magedanz, R.H. Glitho (Eds.), Mobile Agents for Telecommunication Applications. IX, 297 pages. 2003.

Vol. 2880: H.L. Bodlaender (Ed.), Graph-Theoretic Concepts in Computer Science. XI, 386 pages. 2003.

Vol. 2879: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. XXXIV, 1003 pages. 2003.

Vol. 2878: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. XXXIII, 819 pages. 2003.

Vol. 2877: T. Böhme, G. Heyer, H. Unger (Eds.), Innovative Internet Community Systems. VIII, 263 pages. 2003.

Vol. 2876: M. Schroeder, G. Wagner (Eds.), Rules and Rule Markup Languages for the Semantic Web. VII, 173 pages. 2003.

Vol. 2875: E. Aarts, R. Collier, E.v. Loenen, B.d. Ruyter (Eds.), Ambient Intelligence. XI, 432 pages. 2003.

Vol. 2874: C. Priami (Ed.), Global Computing. XIX, 255 pages. 2003.

Vol. 2871: N. Zhong, Z.W. Raś, S. Tsumoto, E. Suzuki (Eds.), Foundations of Intelligent Systems. XV, 697 pages. 2003. (Subseries LNAI).

Vol. 2870: D. Fensel, K.P. Sycara, J. Mylopoulos (Eds.), The Semantic Web - ISWC 2003. XV, 931 pages. 2003.

Vol. 2869: A. Yazici, C. Şener (Eds.), Computer and Information Sciences - ISCIS 2003. XIX, 1110 pages. 2003.

Vol. 2868: P. Perner, R. Brause, H.-G. Holzhütter (Eds.), Medical Data Analysis. VIII, 127 pages. 2003.

Vol. 2866: J. Akiyama, M. Kano (Eds.), Discrete and Computational Geometry. VIII, 285 pages. 2003.

Vol. 2865: S. Pierre, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. X, 293 pages. 2003.

Vol. 2864: A.K. Dey, A. Schmidt, J.F. McCarthy (Eds.), UbiComp 2003: Ubiquitous Computing. XVII, 368 pages. 2003.

Vol. 2863: P. Stevens, J. Whittle, G. Booch (Eds.), "UML" 2003 - The Unified Modeling Language. XIV, 415 pages. 2003.

Vol. 2860: D. Geist, E. Tronci (Eds.), Correct Hardware Design and Verification Methods. XII, 426 pages. 2003.

Vol. 2859: B. Apolloni, M. Marinaro, R. Tagliaferri (Eds.), Neural Nets. X, 376 pages. 2003.

Vol. 2857: M.A. Nascimento, E.S. de Moura, A.L. Oliveira (Eds.), String Processing and Information Retrieval. XI, 379 pages. 2003.

Vol. 2856: M. Smirnov (Ed.), Quality of Future Internet Services. IX, 293 pages. 2003.

Vol. 2855: R. Alur, I. Lee (Eds.), Embedded Software. X, 373 pages. 2003.

Vol. 2854: J. Hoffmann, Utilizing Problem Structure in Planing. XIII, 251 pages. 2003. (Subseries LNAI).

Vol. 2853: M. Jeckle, L.-J. Zhang (Eds.), Web Services - ICWS-Europe 2003. VIII, 227 pages. 2003.

Vol. 2852: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. VIII, 509 pages. 2003.

Vol. 2851: C. Boyd, W. Mao (Eds.), Information Security. XI, 453 pages. 2003.

Vol. 2849: N. García, L. Salgado, J.M. Martínez (Eds.), Visual Content Processing and Representation. XII, 352 pages. 2003.

Vol. 2848: F.E. Fich (Ed.), Distributed Computing. X, 367 pages. 2003.

Vol. 2847: R.d. Lemos, T.S. Weber, J.B. Camargo Jr. (Eds.), Dependable Computing. XIV, 371 pages. 2003.

Vol. 2846: J. Zhou, M. Yung, Y. Han (Eds.), Applied Cryptography and Network Security. XI, 436 pages. 2003.

Vol. 2845: B. Christianson, B. Crispo, J.A. Malcolm, M. Roe (Eds.), Security Protocols. VIII, 243 pages. 2004.

Vol. 2844: J.A. Jorge, N. Jardim Nunes, J. Falcão e Cunha (Eds.), Interactive Systems. Design, Specification, and Verification. XIII, 429 pages. 2003.

Vol. 2843: G. Grieser, Y. Tanaka, A. Yamamoto (Eds.), Discovery Science. XII, 504 pages. 2003. (Subseries LNAI).

Vol. 2842: R. Gavaldá, K.P. Jantke, E. Takimoto (Eds.), Algorithmic Learning Theory. XI, 313 pages. 2003. (Subseries LNAI).

Vol. 2841: C. Blundo, C. Laneve (Eds.), Theoretical Computer Science. XI, 397 pages. 2003.

Contents

1. Introduction

> *"Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it."*
> Eugene C. Freuder, Constraints, April 1997

Overview

Constraint programming is an emergent software technology for declarative description and effective solving of large, particularly combinatorial, problems especially in areas of planning and scheduling. It has recently emerged as a research area that combines researchers from a number of fields, including Artificial Intelligence, Programming Languages, Symbolic Computing and Computational Logic. Constraint networks and constraint satisfaction problems have been studied in Artificial Intelligence starting from the seventies. Systematic use of constraints in programming has started in the eighties. In constraint programming the programming process consists of the generation of requirements (constraints) and solution of these requirements, by specialized constraint solvers.

Constraint programming has been successfully applied in numerous domains. Recent applications include computer graphics (to express geometric coherence in the case of scene analysis), natural language processing (construction of efficient parsers), database systems (to ensure and/or restore consistency of the data), operations research problems (like optimization problems), molecular biology (DNA sequencing), business applications (option trading), electrical engineering (to locate faults) and circuit design (to compute layouts).

This book is centered around the notion of constraint solving [189] and programming [141]. The interest in constraint satisfaction problems can be easily justified, since they represent a very powerful, general, and declarative knowledge representation formalism. In fact, many real-life situations can be faithfully described by a set of objects, together with some constraints among them. Examples of such situations can be found in several areas, like VLSI, graphics, typesetting, scheduling, planning, as well as CAD, decision-support systems and robotics.

1.1 From the Beginning ...

The constraint idea comes from the early 1960's when Sutherland introduced Sketchpad [187], the first interactive graphical interface that solved geometric constraints. After that came Fikes' REF-ARF [97] and at the beginning of 1970's Montanari described fundamental properties of the constraints when applied to picture processing [149]. Another study in finite domain constraint satisfaction was done at the end of 1970's in Laurière's ALICE [133], a system developed to solve prediction/detection problems in geology. After these, several constraint languages have been proposed in the literature: the language of Steele ([184]), CONSTRAINTS [186] of Sussman & Steele, Thinglab ([60]) and Bertrand ([134]).

From Sketchpad until now, a lot of research has been done and improvements made, and the classical constraint satisfaction problems (CSPs) [150, 139] have been shown to be a very expressive and natural formalism to specify many kinds of real-life problems.

1.2 Applications

Today, the use of the constraint programming idea to solve many real-life problem is reality. Many important companies develop tools based on the constraint technology to solve *assignment*, *network management*, *scheduling*, *transport* and many other problems:

1.2.1 Assignment Problems

Assignment problems were one of the first type of industrial applications that were solved with the CLP technology. These problems usually have to handle two types of resources, and constraints among them, and try to assign one resource of the first kind to one of the second kind such that all constraints are satisfied.

An example is the stand allocation for airports, where aircrafts (the first kind of resources) must be parked on the available stands (the second kind of resources) during their stay at the airport. The first industrial CLP application was developed for the HIT container harbor in Hong Kong [161], using the language CHIP: the objective was to allocate berths to container ships in the harbor, in such a way that resources and stacking space for the containers is available. Other Hong Kong applications are at the airport, where a CHIP-based system is used to solve the counter allocation problem [69], and another constraint-based system, which uses the ILOG libraries, is used for the stand allocation problem since mid-1998 [70]. Another system, called APACHE [88], was a demonstrator for stand allocation at Roissy airport in Paris: the objective was to replan the allocation when a change of arrival/departure times occurred.

1.2.2 Personnel Assignment

Personnel assignment problems are a special case of assignment problems where one resource type consists of humans. This peculiarity makes them specific enough to be considered separately. In fact, changing work rules and regulations impose difficult constraints which evolve over time. Also, user preferences