Petko Valtchev
Robert Jäschke (Eds.)

LNAI 6628

Formal Concept Analysis

9th International Conference, ICFCA 2011
Nicosia, Cyprus, May 2011
Proceedings

Springer

Lecture Notes in Artificial Intelligence 6628

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Petko Valtchev Robert Jäschke (Eds.)

Formal Concept Analysis

9th International Conference, ICFCA 2011
Nicosia, Cyprus, May 2-6, 2011
Proceedings

Springer

Volume Editors

Petko Valtchev
Université du Québec à Montréal, Département d'Informatique
C.P.8888, Succ. Centre-Ville, Montréal, Québec H3C 3P8, Canada
E-mail: valtchev.petko@uqam.ca

Robert Jäschke
Universität Kassel, Fachgebiet Wissensverarbeitung
Wilhelmshöher Allee 73, 34121 Kassel, Germany
E-mail: jaeschke@cs.uni-kassel.de

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-20513-2 e-ISBN 978-3-642-20514-9
DOI 10.1007/978-3-642-20514-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011925366

CR Subject Classification (1998): I.2, G.2.1-2, F.4.1-2, D.2.4, H.3

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The present volume features a selection of the papers presented at the 9th International Conference on Formal Concept Analysis (ICFCA 2011). Over the years, the ICFCA conference series has grown into the premier forum for dissemination of research on topics from formal concept analysis (FCA) theory and applications, as well as from the related fields of lattices and partially ordered structures.

FCA is a multi-disciplinary field with strong roots in the mathematical theory of partial orders and lattices, with tools originating in computer science and artificial intelligence. FCA emerged in the early 1980s from efforts to restructure lattice theory to promote better communication between lattice theorists and potential users of lattice-based methods for data management. Initially, the central theme was the mathematical formalization of concept and conceptual hierarchy. Since then, the field has developed into a constantly growing research area in its own right with a thriving theoretical community and an increasing number of applications in data and knowledge processing including disciplines such as data visualization, information retrieval, machine learning, software engineering, data analysis, data mining, social networks analysis, etc.

ICFCA 2011 was held during May 2–6, 2011, in Nicosia, Cyprus. The Program Committee received 49 high-quality submissions that were subjected to a highly competitive selection process. Each paper was reviewed by three referees (exceptionally two or four). After a first round, some papers got a definitive acceptance status, while others got accepted conditionally to improvements in their content. The latter got to a second round of reviewing. The overall outcome was the acceptance of 16 submissions as regular papers for presentation at the conference and publication in this volume. Another seven papers were assessed as valuable for discussion at the conference and were therefore collected in the supplementary proceedings. The regular papers presented hereafter cover advances on a wide range of subjects from FCA and related fields.

A first group of papers tackled mathematical problems within the FCA field. A subset thereof focused on factor identification within the incidence relation or its lattice representation (papers by Glodeanu and by Krupka). The remainder of the group proposed characterizations of particular classes of ordered structures (papers by Doerfel and by Ganter et al.). A second group of papers addressed algorithmic problems from FCA and related fields. Two papers approached their problems from an algorithmic complexity viewpoint (papers by Distel and by Babin and Kuznetsov), while the final paper in this group addressed algorithmic problems for general lattices, i. e., not represented as formal contexts, with an FCA-based approach (work by Balcázar and Tîrnăucă). A third group studied alternative approaches for extending the expressive power of the core FCA, e. g., by generalizing the standard one-valued attributes to attributes valued in

algebraic rings (work by González Calabozo et al.), by introducing pointer-like attributes, a. k. a. links (paper by Kötters), or by substituting set-shaped concept intents with modal logic expressions (paper by Soldano and Ventos). A fourth group focused on data mining-oriented aspects of FCA: agreement lattices in structured data mining (paper by Nedjar et al.), triadic association rule mining (work by Missaoui and Kwuida), and bi-clustering of numerical data (Kaytoue et al.). An addional paper shed some initial light on a key aspect of FCA-based data analysis and mining, i. e., the filtering of interesting concepts (paper by Belohlavek and Macko). Finally, a set of exciting applications of both basic and enhanced FCA frameworks to practical problems were described: in analysis of gene expression data (the already mentioned work by González Calabozo et al.), in Web services composition (paper by Azmeh et al.), and in browsing and retrieval of structured data (work by Wray and Eklund). This volume also contains three keynote papers submitted by the invited speakers of the conference.

All these contributions constitute a volume of high quality which is the result of the hard work done by the authors, the invited speakers and the reviewers. We therefore wish to thank the members of the Program Committee and of the Editorial Board, whose steady involvement and professionalism helped a lot. We would also like to acknowledge the participation of all the external reviewers, who sent many valuable comments. Kudos also go to EasyChair for having made the reviewing/editing process a real pleasure. Special thanks go to the Cyprus Tourism Organisation for sponsoring the conference and to the University of Nicosia for hosting it. Finally, we wish to thank the Conference Chair, Florent Domenach, and his colleagues from the Organizing Committee for the mountains of energy they put behind the conference organization process right from the beginning in order to make it a total success. We would also like to express our gratitude to Dr. Peristianis, President of the University of Nicosia, for his personal support.

May 2011

Petko Valtchev
Robert Jäschke

Organization

The International Conference on Formal Concept Analysis is the annual conference and principal research forum in the theory and practice of Formal Concept Analysis. The inaugural International Conference on Formal Concept Analysis was held at the Technische Universität Darmstadt, Germany, in 2003. Subsequent ICFCA conferences were held at the University of New South Wales in Sydney, Australia, 2004, Université d'Artois, Lens, France, 2005, Institut für Algebra, Technische Universität Dresden, Germany, 2006, Université de Clermont-Ferrand, France, 2007, Université du Québec à Montréal, Canada, 2008, Darmstadt University of Applied Sciences, Germany, 2009, and Agadir, Morocco, 2010. ICFCA 2011 took place at the University of Nicosia, Cyprus. Its committees are listed below.

Executive Committee

Conference Chair

Florent Domenach	University of Nicosia, Cyprus

Conference Organizing Committee

George Chailos	University of Nicosia, Cyprus
Nectarios Papanicolaou	University of Nicosia, Cyprus
George Portides	University of Nicosia, Cyprus
Andreas Savva	University of Nicosia, Cyprus

Conference Proceedings

Program Chairs

Robert Jäschke	Universität Kassel, Germany
Petko Valtchev	Université du Québec à Montréal, Canada

Editorial Board

Peter Eklund	University of Wollongong, Australia
Sebastien Ferré	Université de Rennes 1, France
Bernhard Ganter	Technische Universität Dresden, Germany
Robert Godin	Université du Québec à Montréal (UQAM), Canada
Sergei Kuznetsov	Higher School of Economics, Moscow, Russia

Leonard Kwuida	Zurich University of Applied Sciences, Switzerland
Raoul Medina	LIMOS, Université de Clermont-Ferrand 2, France
Rokia Missaoui	Université du Québec en Outaouais (UQO), Canada
Sergei Obiedkov	Higher School of Economics, Moscow, Russia
Uta Priss	Napier University, Edinburgh, UK
Sebastian Rudolph	AIFB, University of Karlsruhe, Germany
Stefan E. Schmidt	Technische Universität Dresden, Germany
Baris Sertkaya	SAP Research Center, Dresden, Germany
Gerd Stumme	University of Kassel, Germany
Rudolf Wille	Technische Universität Darmstadt, Germany
Karl Erich Wolff	University of Applied Sciences, Darmstadt, Germany

Program Committee

Mike Bain	University of New South Wales, Sydney, Australia
Jaume Baixeries	Polytechnical University of Catalonia, Spain
Peter Becker	The University of Queensland, Brisbane, Australia
Radim Belohlavek	Binghamton University - State University of New York, USA
Sadok Ben Yahia	Faculty of Sciences of Tunis, Tunisia
Jean-François Boulicaut	INSA Lyon, France
Claudio Carpineto	Fondazione Ugo Bordoni, Italy
Nathalie Caspard	LACL, Université Paris 12, France
Frithjof Dau	SAP Research CEC Dresden, Germany
Stephan Doerfel	KDE Group, University of Kassel, Germany
Vincent Duquenne	ECP6-CNRS, Université Paris 6, France
Alain Gély	Université Paul Verlaine, Metz, France
Wolfgang Hesse	Philipps-Universität Marburg, Germany
Marianne Huchard	LIRMM, Université Montpellier, France
Tim B. Kaiser	SAP AG, Germany
Derrick G. Kourie	University of Pretoria, South Africa
Markus Krötzsch	Universität Karlsruhe, Germany
Marzena Kryszkiewicz	Warsaw University of Technology, Poland
Wilfried Lex	Universität Clausthal, Germany
Lotfi Lakhal	Aix-Marseille Université, France
Bruno Leclerc	CAMS-EHESS, Paris, France
Engelbert Mephu Nguifo	LIMOS, Université de Clermont Ferrand 2, France
Amedeo Napoli	LORIA, Nancy, France

Lhouari Nourine	LIMOS, Université de Clermont Ferrand 2, France
Jean-Marc Petit	LIRIS, INSA Lyon, France
Alex Pogel	New Mexico State University, Las Cruces, USA
Sandor Radeleczki	University of Miskolc, Hungary
Camille Roth	CNRS/EHESS, Paris, France
Mohamed Rouane-Hacene	Université du Québec à Montréal (UQAM), Canada
Jürg Schmid	Universität Bern, Switzerland
Andreja Tepavčević	University of Novi Sad, Serbia

External Reviewers

Alain Casali	Aix-Marseille Université, France
Laurent Brehelin	LIRMM, Université Montpellier, France
Peggy Cellier	Université de Rennes 1, France
Philippe Fournier-Viger	National Cheng Kung University, Taiwan
Tarek Hamrouni	University of Tunis, Tunisia
Sébastien Nedjar	Aix-Marseille Université, France
Yoan Renaud	LIRIS, INSA Lyon, France
Chris Schulz	University of Pretoria, South Africa
Laszlo Szathmary	Université du Québec à Montréal, Canada

Sponsoring Institutions

University of Nicosia, Cyprus
Cyprus Tourism Organisation

Table of Contents

Invited Talks

Regular Contributions

Inductive Databases and Constraint-Based Data Mining

Sašo Džeroski

Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract. We briefly introduce the notion of an inductive database, explain its relation to constraint-based data mining, and illustrate it on an example. We then discuss constraints and constraint-based data mining in more detail. We further give an overview of recent developments in the area, focusing on those made within the IQ project and presented in a recent volume with the same title as this paper, edited by the author, Bart Goethals and Panče Panov, and published by Springer.

1 Inductive Databases

Inductive databases (IDBs, (Imielinski and Mannila 1996, De Raedt 2002a)) are an emerging research area at the intersection of data mining and databases. Inductive databases contain both data and patterns (in the broader sense, which includes frequent patterns, predictive models, and other forms of generalizations). IDBs embody a database perspective on knowledge discovery, where knowledge discovery processes become query sessions. KDD thus becomes an extended querying process (Imielinski and Mannila 1996) in which both the data and the patterns that hold (are valid) in the data are queried.

Roughly speaking, an *inductive database* instance contains: (1) Data (e.g., a relational database, a deductive database), (2) Patterns (e.g., itemsets, episodes, subgraphs, substrings, ...), and (3) Models (e.g., classification trees, regression trees, regression equations, Bayesian networks, mixture models, ...). The difference between patterns (such as frequent itemsets) and models (such as regression trees) is that patterns are *local* (they typically describe properties of a subset of the data), whereas models are *global* (they characterize the entire data set). Patterns are typically used for descriptive and models for predictive purposes.

A *query language* for an inductive database is an extension of a database query language that allows us to: (1) select, manipulate and query data in the database as in current DBMSs, (2) select, manipulate and query "interesting" patterns and models (e.g., patterns that satisfy *constraints* w.r.t. frequency, generality, etc. or models that satisfy *constraints* w.r.t. accuracy, size, etc.), and (3) *match* patterns or models with data, e.g., select the data in which some patterns hold, or predict a property of the data with a model.

Inductive Databases and Queries: An Example. To clarify what is meant by the terms inductive database and inductive query, we illustrate them by an example from the area of bio-/chemo-informatics. Consider the task of

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 1–17, 2011.

discovering a model that predicts whether chemical compounds are toxic or not. In this context, the data part of the IDB will consist of one or more sets of compounds. In our illustration below, there are two sets: the *active* (toxic) and the *inactive* (non-toxic) compounds. Assume, furthermore, that for each of the compounds, the two dimensional (i.e., graph) structure of their molecules is represented within the database, together with a number of attributes that are related to the outcome of the toxicity tests. The database query language of the IDB will allow the user (say a predictive toxicology scientist) to retrieve information about the compounds (i.e., their structure and properties). The inductive query language will allow the scientist to generate, manipulate and apply patterns and models of interest.

As a first step towards building a predictive model, the scientist may want to find local patterns (in the form of compound substructures or molecular fragments), that are "interesting", i.e., satisfy certain constraints. An example inductive query may be written as follows: $F = \{\tau | (\tau \in AZT) \wedge (freq(\tau, Active) \geq 15\%) \wedge (freq(\tau, Inactive) \leq 5\%)\}$. This should be read as: "Find all molecular fragments that appear in the compound AZT (which is a drug for AIDS), occur frequently in the active compounds ($\geq 15\%$ of them) and occur infrequently in the inactive ones ($\leq 5\%$ of them)".

Once an interesting set of patterns has been identified, they can be used as descriptors (attributes) for building a model (e.g., a decision tree that predicts activity). A data table can be created by first constructing one feature/column for each pattern, then one example/row for each data item. The entry at a given column and row has value "true" if the corresponding pattern (e.g., fragment) appears in the corresponding data item (e.g., molecule). The table could be created using a traditional query in a database query language, combined with IDB matching primitives.

Suppose we have created a table with columns corresponding to the molecular fragments F returned by the query above and rows corresponding to compounds in $Active \bigcup Inactive$, and we want to build a global model (decision tree) that distinguishes between active and inactive compounds. The toxicologist may want to constrain the decision tree induction process, e.g., requiring that the decision tree contains at most k leaves, that certain attributes are used before others in the tree, that the internal tests split the nodes in (more or less) proportional subsets, etc. She may also want to impose constraints on the accuracy of the induced tree.

Note that in the above scenario, a sequence of queries is used. This requires that the *closure property* be satisfied: the result of an inductive query on an IDB instance should again be an IDB instance. Through supporting the processing of sequences of inductive queries, IDBs would support the entire KDD process, rather than individual data mining steps.

Inductive Queries and Constraints. In inductive databases (Imielinski and Mannila 1996), patterns become "first-class citizens" and can be stored and manipulated just like data in ordinary databases. Ordinary queries can be used to access and manipulate data, while inductive queries (IQs) can be

used to generate (mine), manipulate, and apply patterns. KDD thus becomes an extended querying process in which both the data and the patterns that hold (are valid) in the data are queried. In IDBs, the traditional KDD process model where steps like pre-processing, data cleaning, and model construction follow each other in succession, is replaced by a simpler model in which all operations (pre-processing, mining, post-processing) are queries to an IDB and can be interleaved in many different ways.

Given an IDB that contains data and patterns (or other types of generalizations, such as models), several different types of queries can be posed. Data retrieval queries use only the data and their results are also data: no pattern is involved in the query. In IDBs, we can also have cross-over queries that combine patterns and data in order to obtain new data, e.g., apply a predictive model to a dataset to obtain predictions for a target property. In processing patterns, the patterns are queried without access to the data: this is what is usually done in the post-processing stages of data mining. Inductive (data mining) queries use the data and their results are patterns (generalizations): new patterns are generated from the data: this corresponds to the traditional data mining step.

A general statement of the problem of data mining (Mannila and Toivonen 1997) involves the specification of a language of patterns (generalizations) and a set of constraints that a pattern has to satisfy. The constraints can be language constraints and evaluation constraints: The first only concern the pattern itself, while the second concern the validity of the pattern with respect to a given database. Constraints thus play a central role in data mining and constraint-based data mining (CBDM) is now a recognized research topic (Bayardo 2002). The use of constraints enables more efficient induction and focusses the search for patterns on patterns likely to be of interest to the end user.

In the context of IDBs, inductive queries consist of constraints. Inductive queries can involve language constraints (e.g., find association rules with item A in the head) and evaluation constraints, which define the validity of a pattern on a given dataset (e.g., find all item sets with support above a threshold or find the 10 association rules with highest confidence).

Different types of data and patterns have been considered in data mining, including frequent itemsets, episodes, Datalog queries, and graphs. Designing inductive databases for these types of patterns involves the design of inductive query languages and solvers for the queries in these languages, i.e., CBDM algorithms. Of central importance is the issue of defining the primitive constraints that can be applied for the chosen data and pattern types, that can be used to compose inductive queries. For each pattern domain (type of data, type of pattern, and primitive constraints), a specific solver is designed, following the philosophy of constraint logic programming (De Raedt 2002b).

The Promise of Inductive Databases. While knowledge discovery in databases (KDD) and data mining have enjoyed great popularity and success over the last two decades, there is a distinct lack of a generally accepted framework for data mining (Fayyad et al. 2003). In particular, no framework exists that

can elegantly handle simultaneously the mining of complex/structured data, the mining of complex (e.g., relational) patterns and use of domain knowledge, and support the KDD process as a whole, three of the most challenging/important research topics in data mining (Yang and Wu 2006).

The IDB framework is an appealing approach towards developing a generally accepted framework/theory for data mining, as it employs declarative queries instead of ad-hoc procedural constructs: Namely, in CBDM, the conditions/constraints that a pattern has to satisfy (to be considered valid/interesting) are stated explicitly and are under direct control of the user/data miner. The IDB framework holds the promise of facilitating the formulation of an "algebra" for data mining, along the lines of Codd's relational algebra for databases (Calders et al. 2006b, Johnson et al. 2000).

Different types of structured data have been considered in CBDM. Besides itemsets, other types of frequent/local patterns have been mined under constraints, e.g., on strings, sequences of events (episodes), trees, graphs and even in a first-order logic context (patterns in probabilistic relational databases). More recently, constraint-based approaches to structured prediction have been considered, where models (such as tree-based models) for predicting hierarchies of classes or sequences / time series are induced under constraints.

Different types of local patterns and global models have been considered as well, such as rule-based predictive models and tree-based clustering models. When learning in a relational setup, background / domain knowledge can be naturally taken into account (through additional relations/predicates). Also, the constraints provided by the user in CBDM can be viewed as a form of domain knowledge that focuses the search for patterns / model towards interesting and useful ones.

The IDB framework is also appealing for data mining applications, as it supports the entire KDD process (Boulicaut et al. 1999). In inductive query languages, the results of one (inductive) query can be used as input for another. Nontrivial multi-step KDD scenarios can be thus supported in IDBs, rather than just single data mining operations.

2 Constraint-Based Data Mining

"Knowledge discovery in databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data", state Fayyad et al. (1996). According to this definition, data mining (DM) is the central step in the KDD process concerned with applying computational techniques (i.e., data mining algorithms implemented as computer programs) to actually find patterns that are valid in the data. In constraint-based data mining (CBDM), a pattern/model is valid if it satisfies a set of constraints.

The basic concepts/entities of data mining include data, data mining tasks, and generalizations (e.g., patterns and models). The validity of a generalization on a given set of data is related to the data mining task considered. Below we briefly discuss the basic entities of data mining and the task of CBDM.

2.1 Basic Data Mining Entities

Data. A data mining algorithm takes as input a set of data. An individual datum in the data set has its own structure, e.g., consists of values for several attributes, which may be of different types or take values from different ranges. We assume all data items are of the same type (and share the same structure).

More generally, we are given a data type T and a set of data D of this type. It is of crucial importance to be able to deal with structured data, as these are attracting an ever increasing amount of attention within data mining. The data type T can thus be an arbitrarily complex data type, composed from a set of basic/primitive types (such as Boolean and Real) by using type constructors (such as Tuple, Set or Sequence).

Generalizations. We will use the term generalization to denote the output of different data mining tasks, such as pattern mining, predictive modeling and clustering. Generalizations will thus include probability distributions, patterns (in the sense of frequent patterns), predictive models and clusterings. All of these are defined on a given type of data, except for predictive models, which are defined on a pair of data types. Note that we allow arbitrary (arbitrarily complex) data types. The typical case in data mining considers a data type T = Tuple(T_1, ..., $_k$), where each of T_1, ..., T_k is Boolean, Discrete or Real.

We will discuss briefly here local patterns and global models (predictive models and clusterings). Note that both are envisaged as first-class citizens of inductive databases. More detailed discussions of all types of generalizations are given by Dzeroski (2007).

A *pattern P on type T* is a Boolean function on objects of type T: A pattern on type T is true or false on an object of type T. We restrict the term pattern here to the sense that it is most commonly used, i.e., in the sense of frequent pattern mining. A *predictive model M for types T_d, T_c* is a function that takes an object of type T_d (description) and returns one of type T_c (class/target). We allow both T_d and T_c to be arbitrarily complex data types, with classification and regression as special cases (when T_c has nominal, respectively numeric values). A *clustering C on a set of objects S of type T* is a function from S to $\{1, \ldots, k\}$, where k is the number of clusters (with $k \leq |S|$). It partitions a set of objects into subsets called clusters by mapping each object to a cluster identifier.

Data Mining Tasks. In essence, the task of data mining is to produce a generalization from a given set of data. A plethora of data mining tasks has been considered so far in the literature, with four covering the majority of data mining research: approximating the (joint) probability distribution, clustering, learning predictive models, and finding valid (frequent) patterns. We will focus here on the last two of these.

In *learning a predictive model*, we are given a dataset consisting of example input/output pairs (d, c), where each d is of type T_d and each c is of type T_c. We want to find a model m (mapping from T_d to T_c), for which the observed and predicted outputs, i.e., c and $\hat{c} = m(d)$, match closely. In *pattern discovery*, the task is to find all local patterns from a given pattern language (class) that

satisfy the required conditions. A prototypical instantiation of this task is the task of finding frequent itemsets (sets of items, such as $\{bread, butter\}$), which occur frequently (in a sufficiently high proportion) in a given set of transactions (market baskets) (Aggrawal et al. 1993). In *clustering*, we are given a set of examples (object descriptions), and the task is to partition these examples into subsets, called clusters. The notion of a distance (or conversely, similarity) is crucial here: The goal of clustering is to achieve high similarity between objects within a cluster (intra-cluster similarity) and low similarity between objects from different clusters (inter-cluster similarity).

2.2 The Task(s) of (Constraint-Based) Data Mining

Having set the scene, we can now attempt to formulate a very general version of the problem addressed by data mining. We are given a dataset D, consisting of objects of type T. We are also given a data mining task, such as learning a predictive model or pattern discovery. We are further given C_G a family/class of generalizations (patterns/models), such as decision trees, from which to find solutions to the data mining task at hand. Finally, a set of constraints C is given, concerning both the syntax (form) and semantics (validity) that the generalizations have to satisfy.

The problem addressed by constraint-based data mining (CBDM) is to find a set of generalizations G from C_G that satisfy the constraints in C: A desired cardinality on the solution set is usually specified.

In the above formulation, all of data mining is really constraint-based. We argue that the 'classical' formulations of and approaches to data mining tasks, such as clustering and predictive modelling, are a special case of the above formulation. A major difference between the 'classical' data mining paradigm and the 'modern' constraint-based one is that the former typically considers only one quality metric, e.g., minimizes predictive error or intra-cluster variance, and produces only one solution (predictive model or clustering).

A related difference concerns the fact that most of the 'classical' approaches to data mining are heuristic and do not give any guarantees regarding the solutions. For example, a decision tree generated by a learning algorithm is typically not guaranteed to be the smallest or most accurate tree for the given dataset. On the other hand, CBDM approaches have typically been concerned with the development of so-called 'optimal solvers', i.e., data mining algorithms that return the complete set of solutions that satisfy a given set of constraints or the k best solutions (e.g., the k itemsets with highest correlation to a given target).

3 The Anatomy of Constraints

Constraints in CBDM are propositions/statements about generalizations (e.g., patterns or models). In the most basic setting, the propositions are either true or false (Boolean valued): If true, the generalization satisfies the constraint. In CBDM, we are seeking generalizations that satisfy a given set of constraints.

3.1 Types of Constraints

Many types of constraints are currently used in CBDM, which can be divided along several dimensions. Along the first dimension, we distinguish between primitive and composite constraints. Along the second dimension, we distinguish between language and evaluation constraints. Along the third dimension, we have Boolean (or hard), soft, and optimization constraints.

Primitive and Composite Constraints. Recall that constraints in CBDM are propositions on generalizations. Some of these propositions are atomic in nature (and are not decomposable into simpler propositions). In mining frequent itemsets, the constraints "item *bread* must be contained in the itemset" and "the itemsets should have a frequency higher than 10" are atomic/primitive.

Primitive constraints can be combined by using boolean operators, i.e., negation, conjunction and disjunction. The resulting constraints are called composite constraints. The properties of the composite constrains (such as monotonicity/ anti-monotonicity discussed below) depend on the properties of the primitive constraints and the operators used to combine them.

Language and Evaluation Constraints. Constraints typically refer to either the form / syntax of generalizations or their semantics / validity with respect to the data. In the first case, they are called language constraints, and in the second evaluation constraints. Below we discuss primitive language and evaluation constraints. Note that these can be used to form composite language constraints, composite evaluation constraints, and composite constraints that mix language and evaluation primitives.

Language constraints concern the syntax / representation of a pattern/model, i.e., refer only to its form. We can check whether they are satisfied or not without accessing the data that we have been given as a part of the data mining task. If we are in the context of inductive databases and queries, post-processing queries on patterns / models are composed of language constraints.

A commonly used type of language constraints is that of subsumption constraints. For example, we might be interested in finding frequent itemsets where a specific item, e.g., *beer*, occurs (that is itemsets that subsume *beer*). Another type of language constraints involves (cost) functions on patterns / models. An example of these is the size of a decision tree: We can look for decision trees of at most ten nodes. The cost functions (such as size) discussed here are mappings from the representation of a pattern/model to non-negative reals: Boolean (hard) language constraints put thresholds on the values of these functions.

Evaluation constraints concern the semantics of patterns / models, in particular as applied to a given set of data. Evaluation constraints typically involve evaluation functions, comparing them to constant thresholds. Evaluation functions measure the validity of patterns/models on a given set of data: They take as input a pattern or a model and a set of data, returning a real value as output. For example, the frequency of a pattern on a given dataset is an evaluation function, as is the classification error of a predictive model. Evaluation constraints

typically compare the value of an evaluation function to a constant threshold, e.g., minimum support or maximum error.

Hard, Soft and Optimization Constraints. *Hard constraints* in CBDM are Boolean functions on patterns / models. This means that a constraints is either satisfied or not satisfied. Constraints define what patterns are valid or interesting: The fact that interestingness is not a dichotomy (Bistarelli and Bonchi 2005) has lead to the introduction of so-called soft constraints.

Soft constraints do not dismiss a pattern for violating a constraint; rather, the pattern incurring a penalty for violating a constraint. In the cases where we typically consider a larger number of binary constraints, such as must-link and cannot-link constraints in constrained clustering (Wagstaff and Cardie 2000), a fixed penalty may be assigned for violating each constraint. In the soft constraint setting, all patterns/models are solutions to a different degree: Patterns with lower penalty satisfy the constraints better (to a higher degree) and we look for patterns with minimum penalty.

Optimization constraints allow us to ask for (a fixed-size set of) patterns/ models that have a maximal/ minimal value for a given cost or evaluation function. Example queries with such constraints could ask for the k most frequent itemsets or the top k correlated patterns. We might also ask for the most accurate decision tree of size five, or the smallest tree at least 90% accurate.

3.2 Functions Used in Constraints

As discussed above, functions are used to compose constraints. Language constraints use language cost functions, while evaluation constraints use evaluation functions. An mportant property of such functions is monotonicity, and a property of patterns related to monotonicity is closedness.

Language Cost Functions. The cost functions that are used in language constraints concern the representation of generalizations (patterns/models/...). Most often, these functions are related to the size/complexity of the representation. They are different for different classes of generalizations, e.g., for itemsets, mixture models of Gaussians, linear models or decision trees. For itemsets, the size can be the cardinality of the itemset; for decision trees, the total number of nodes; and for linear models, the number of variables (with non-zero coefficients).

More general versions of cost functions involve costs of the individual language elements, such as items or attributes, and sum/aggregate these over all elements appearing in the pattern/model. These are motivated by practical considerations, e.g., costs for items in an itemset and total cost of a market basket, or the cost of lab tests in medical predictive models. Language constraints as commonly used in CBDM involve thresholds on the values of cost functions (e.g., find a decision tree of size at most ten leaves).

Evaluation Functions. The evaluation functions used in evaluation constraints are tightly coupled with the data mining task at hand. If we are solving a predictive modelling problem, the evaluation function used will most likely concern

predictive error. If we are solving a frequent pattern mining problem, the evaluation function used will definitely concern the frequency of the patterns.

For the task of *pattern discovery*, with the discovery of frequent patterns as the prototypical instantiation, the primary evaluation function is frequency. Recall that patterns are Boolean functions, assigning values of true or false to data points. The frequency of a pattern on a set of data is the cardinality (or the proportion) of the subset of the data for which the pattern is true.

For *predictive models*, predictive error is the function typically used in constraints. The error function used crucially depends on the type of the target predicted. For a discrete target (classification), misclassification error/cost can be used; for a continuous target (regression), mean absolute error can be used. In general, for a target of type T_c, we can use the mean error defined by a distance (or cost) function d_c between objects of type T_c.

Similar evaluation functions can be defined for *probabilistic predictive modeling*, a subtask of predictive modeling. For the data mining task of **clustering**, the quality of a clustering is typically evaluated with intra-cluster variance (ICV) in partition-based clustering. For density-based clustering, a variant of the task of *estimating the probability distribution*, scoring functions for distributions/ densities are used, typically based on likelihood or log-likelihood (Hand et al. 2001).

Scoring functions. Language constraints are often combined with evaluation constraints. The latter can be hard constraints based on thresholds (e.g., find a tree of size at most 10 with classification error of at most 10%) or optimization constraints (e.g., find a tree of size at most 10 and the smallest classification error). Also, optimization constraints may use language-related cost functions, e.g., find the smallest decision tree with classification error lower than 10%.

In the 'classical' formulations of and approaches to data mining tasks, scoring functions often combine evaluation functions and language cost functions. The typical score function is a linear combination of the two, i.e., $Score(G, D) = w_E \times Evaluation(G.function, D) + w_L \times LanguageCost(G.data)$, where G is the generalization (pattern/model) scored and D is the underlying dataset. For predictive modelling, this can translate to $Score = w_E \times Error + w_S \times Size$.

Monotonicity. The notion of monotonicity of an evaluation (or cost) function on a class of generalizations is often considered in CBDM. In mathematics, a function $f(x)$ is monotonic (monotonically increasing) if $\forall x, y : x < y \rightarrow f(x) \leq f(y)$, i.e., the function preserves the $<$ order. If the function reverses the order, i.e., $\forall x, y : x < y \rightarrow f(x) \geq f(y)$, we call it monotonically decreasing.

In data mining, in addition to the order on Real numbers, we also have a generality order on the class of generalizations. The latter is typically induced by a refinement operator. We say that $g_1 \leq_{ref} g_2$ if g_2 can be obtained from g_1 through a sequence of refinements (and thus g_1 is more general than g_2): we will refer to this order as the refinement order.

An evaluation (or cost) function is called monotonic if it preserves the refinement order or anti-monotonic if it reverses it. More precisely, an evaluation

function f is called monotonic if $\forall g_1, g_2 : g_1 \leq_{ref} g_2 \rightarrow f(g_1) \leq f(g_2)$ and anti-monotonic (or monotonically decreasing) if $\forall g_1, g_2 : g_1 \leq_{ref} g_2 \rightarrow f(g_1) \geq f(g_2)$.

Note that the above notions are defined for both evaluation functions / constraints and for language cost functions / constraints. In this context, the frequency of itemsets is anti-monotonic (it decreases monotonically with the refinement order). The total cost of an itemset and the total prediction cost of a decision tree, on the other hand, are monotonic.

In the CBDM literature (Boulicaut and Jeudy 2005), the refinement order considered is typically the subset relation on itemsets ($\leq_{ref}$ is identical to $\subseteq$). A constraint C (taken as a Boolean function) is considered monotonic if $i_1 \leq_{ref} i_2 \wedge C(i_1)$ implies $C(i_2)$. A maximum frequency constraint of the form $freq(i) \leq \theta$, where θ is a constant, is monotonic. Similarly, minimum frequency/support constraints of the form $freq(i) \geq \theta$, the ones most commonly considered in data mining, are anti-monotonic. A disjunction or a conjunction of anti-monotonic constraints is an anti-monotonic constraint. The negation of a monotonic constraint is anti-monotonic and vice versa.

The notions of monotonicity and anti-monotonicity are important because they allow for the design of efficient CBDM algorithms. Anti-monotonicity means that when a pattern does not satisfy a constraint C, then none of its refinements can satisfy C. It thus becomes possible to prune huge parts of the search space which can not contain interesting patterns. This has been studied within the learning as search framework (Mitchell 1982) and the generic levelwise algorithm from (Mannila and Toivonen 1997) has inspired many algorithmic developments.

Closedness. Finally, let us mention the notion of closedness. A pattern (generalization) is closed, with respect to a given refinement operator $\leq_{ref}$ and evaluation function f, if refining the pattern in any way decreases the value of the evaluation function. More precisely, x is closed if $\forall y, x \leq_{ref} y : f(y) < f(x)$. This notion has primarily been considered in the context of mining frequent itemsets, where a refinement adds an item to an itemset and the evaluation function is frequency. There it plays an important role in condensed representations (Calders et al. 2005). However, it can be defined analogously for other types of patterns.

4 Inductive Queries for Mining Patterns and Models

A wide variety of research on IDBs and queries, as well as CBDM, was conducted within two EU-funded projects. The first (contract number FP5-IST 26469) took place from 2001 to 2004 and was titled cInQ (consortium on discovering knowledge with Inductive Queries). The second (contract number FP6-IST 516169) took place from 2005 to 2008 and was titled IQ (Inductive Queries for mining patterns and models). This section briefly summarizes the research on IDBs and CBDM conducted within the IQ project: A more detailed account can be found in the volume with the same title as this paper, edited by Dzeroski et al. (2010).

4.1 Background

The notions of inductive databases and queries were introduced by Imielinski and Mannila (1996). The notion of constraint-based data mining (CBDM) appears in the data mining literature for the first time towards the end of the 20th century (Han et al. 1999). A special issue of the SIGKDD Explorations bulletin devoted to constraints in data mining was edited by Bayardo (2002).

Following these early efforts, the project cInQ made substantial advances in this area. A detailed overview of the results of the cInQ project is given by Boulicaut et al. (2005). The major contributions of the project, however, can be briefly summarized as follows.

First, an important theoretical framework was introduced for local/frequent pattern mining (e.g., itemsets, strings) under constraints (see, e.g., (De Raedt 2002a)), in which arbitrary boolean combinations of monotonic and anti-monotonic primitives can be used to specify the patterns of interest. Second, major progress was achieved in the area of condensed representations that compress/condense sets of solutions to inductive queries (see, e.g., (Boulicaut et al. 2003)) enabling one to mine dense and/or highly correlated transactional data sets, such as WWW usage data or boolean gene expression data, that could not be mined before. Third, cInQ studied the incorporation of inductive queries for frequent patterns and association rules, in query languages such as SQL and XQuery, also addressing inductive query evaluation and optimization in this context (Meo 2003). Finally, the various approaches to mining sets of (frequent) patterns were successfully used in real-life applications in bio- and chemo-informatics, most notably for finding frequent patterns in molecules (Kramer et al. 2001) and in gene expression data (Becquet et al. 2002).

However, many limitations of IDBs/queries and CBDM remained to be addressed at the end of the cInQ project. Most existing approaches to inductive querying and CBDM focused on mining local patterns for a specific type of data (such as itemsets) and a specific set of constraints (based on frequency-related primitives). Inductive querying of global models, such as mining predictive models or clusterings under constraints remained largely unexplored. Although some integration of frequent pattern mining into database query languages was attempted, most inductive querying/CBDM systems worked in isolation and were not integrated with other data mining tools. Consequently, applications of IDBs and CBDM to practically important problems remained limited.

The IQ project set out to address these challenges to IDBs and CBDM remaining at the end of the cInQ project. The overall goal of the IQ project was to develop a sound theoretical understanding of inductive querying that would enable us to develop effective inductive database systems and to apply them on significant real-life applications. To realize this aim, it had to develop the required theory, representations and primitives for local pattern and global model mining, and integrate these into inductive querying systems, inductive database systems and query languages, and general frameworks for data mining. Based on these advances, it aimed to developed significant show-case applications of inductive querying in the area of bioinformatics.

4.2 Major Results of the IQ Project

In sum, the IQ project has made major progress in several directions. In the first instance, these include further developments in constraint-based mining of frequent patterns, as well as advances in mining global models (predictive models and clusterings) under constraints. At another level, approaches for mining frequent patterns have been integrated with the mining of predictive models (classification) and clusterings (bi-clustering or co-clustering) under constraints. In the quest for integration, inductive query languages, inductive database systems and frameworks for data mining in general have been developed. Finally, applications in bioinformatics which use these advances have been developed.

Advances in mining frequent patterns have been made along several dimensions, including the generalization of the notion of closed patterns. First, the one-dimensional (closed sets) and two-dimensional (formal concepts) cases have been lifted to the case of n-dimensional binary data (Cerf et al. 2008, 2010). Second, the notion of closed patterns (and the related notion of condensed representations) have been extended to the case of multi-relational data (Garriga et al. 2007). Third, and possibly most important, a unified view on itemset mining under constraints has been formulated (De Raedt et al. 2008, Besson et al. 2010) where a highly declarative approach is taken. Most of the constraints used in itemset mining can be reformulated as sets or reified summation constraints, for which efficient solvers exist in constraint programming. This means that, once the constraints have been appropriately formulated, there is no need for special purpose CBDM algorithms.

Additional contributions in mining frequent patterns include the mining of patterns in structured data, fault-tolerant approaches for mining frequent patterns and randomization approaches for evaluating the results of frequent pattern mining. New approaches have been developed for mining frequent substrings in strings (cf. (Rigotti et al. 2010)), frequent paths, trees, and graphs in graphs (cf., e.g., (Bringmann et al. 2006, 2010)), and frequent multi-relational patterns in a probabilistic extension of Prolog named ProbLog (cf. (De Raedt et al. 2010)). Fault-tolerant approaches have been developed to mining bi-sets or formal concepts (cf. (Besson et al. 2010)), as well as string patterns (cf. (Rigotti et al. 2010)): The latter has been used to to discover putative transcription factor binding sites in gene promoter sequences. A general approach to the evaluation of data mining results, including those of mining frequent patterns, has been developed: The approach is based on swap randomization (Gionis et al. 2006).

Advances in mining global models for prediction and clustering have been made along two major directions. The first direction is based on predictive clustering, which unifies prediction and clustering, and can be used to build predictive models for structured targets (tuples, hierarchies, time series). Constraints related to prediction (such as maximum error bounds), as well as clustering (such as must-link and cannot link constraints), can be addressed in predictive clustering trees (Struyf and Dzeroski 2010). Due to its capability of predicting structured outputs, this approach has been successfully used for applications

such as gene function prediction (Vens et al. 2010) and gene expression data analysis (Slavkov and Džeroski 2010).

The second direction is based on integrated mining of (frequent) local patterns and global models (for prediction and clustering). For prediction, the techniques developed range from selecting relevant patterns from a previously mined set as new features of the data, over inducing pattern-based rule sets, to integrating pattern mining and model construction (Bringmann et al. 2010). For clustering, approaches have been developed for constrained clustering by using local patterns as features for a clustering process, computing co-clusters by post-processing collections of local patterns, and using local patterns to characterize given co-clusters (cf., e.g., (Pensa et al. 2008)).

Finally, algorithms have also been developed for constrained prediction and clustering that do not belong to the above two paradigms. These include algorithms for constrained induction of polynomial equations for multi-target prediction (Peckov et al. 2007). A large body of work has been devoted to developing methods for the segmentation of sequences, which can be viewed as a form of constrained clustering (Bingham 2010), where the constraints relate the segments to each other and make the end result more interpretable for the human eye, and/or make the computational task simpler. The segmentation methods have been used to segment genomic sequences.

Advances in integration approaches have been made concerning inductive query languages, inductive database systems and frameworks for data mining based on the notions of IDBs and queries, as well as CBDM. Several inductive query languages have been proposed within the project, such as IQL (Nijssen and De Raedt 2007), which is an extension of the tuple relational calculus with functions, a typing system and various primitives for data mining. IQL is expressive enough to support the formulation of non trivial KDD scenarios, e.g., the formal definition of a typical feature construction phase based on frequent pattern mining followed by a decision tree induction phase.

An inductive database system coming out of the IQ project is embodied within the MiningViews approach (Calders et al. 2006a, Blockeel et al. 2010b). This approach uses the SQL query language to access data, patterns (e.g., frequent itemsets) and models (e.g., decision trees): The patterns/models are stored in a set of relational tables, called mining views, which virtually represent the complete output of the respective data mining tasks. In reality, the mining views are empty and the database system finds the required tuples only when they are queried by the user: This is done by extracting constraints from the SQL queries accessing the mining views and calling an appropriate CBDM algorithm.

A special purpose type of inductive database are experiment databases (Vanschoren and Blockeel 2010): These are databases designed to collect the details of data mining (machine learning) experiments, which run different data mining algorithms on different datasets and tasks, and their results. Like all IDBs, experiment databases store the results of data mining: They store information on datasets, learners, and models resulting from running those learners on those datasets: The datasets, learners and models are described in terms

of predefined properties, rather than being stored in their entirety. A typical IDB stores one datasets and the generalizations derived from them (complete patterns/model), while experiment databases store summary information on experiments concerning multiple datasets. Inductive queries on experiment databases analyze the descriptions of datasets and models, as well as experimental results, in order to find possible relationships between them: In this context, meta-learning is well-supported.

Several proposals of frameworks for data mining were considered within the project, such as the data mining algebra of Calders et al. (2006b). Among these, the general framework for data mining proposed by Dzeroski (2007) defines precisely and formally the basic concepts (entities) in data mining, which are used to frame this chapter. The framework has also served as the basis for developing OntoDM, an ontology of data mining (Panov et al. 2010): While a number of data mining ontologies have appeared recently, the unique advantages of OntoDM include the facts that (a) it is deep, (b) it follows best practices from ontology design and engineering (e.g., small number of relations, alignment with top-level ontologies), and (c) it covers structured data, different data mining tasks, and IDB/CBDM concepts, all of which are orthogonal dimensions that can be combined in many ways.

On the **theory** front, the most important contributions (selected from the above) are as follows. Concerning frequent patterns, they include the extensions of the notion of closed patterns to the case of n-dimensional binary data and multi-relational data and the unified view on itemset mining under constraints in a constraint programming setting. Concerning global models, they include advances in predictive clustering, which unifies prediction and clustering and can be used for structured prediction, as well as advances in integrated mining of (frequent) local patterns and global models (for prediction and clustering). Finally, oncerning integration, they include the MiningViews approach and the general framework/ontology for data mining.

On the **applications** front, the tasks of drug design, gene expression data analysis, gene function prediction, and genome segmentation were considered. In drug design, the more specific task of QSAR (quantitative structure-activity relationships) modeling was addressed: The topic is treated by King et al. (2010). Several applications in gene expression data analysis are discussed by Slavkov and Džeroski (2010). In addition, human SAGE gene expression data have been analyzed (Blachon et al. 2007), where frequent patterns are found first (in a fault-tolerant manner), clustered next, and the resulting clusters (called also quasi-synexpression groups) are then explored by domain experts, making it possible to formulate very relevant biological hypotheses.

Gene function prediction was addressed for several organisms, a variety of datasets, and two annotation schemes (including the Gene Ontology): This application area is discussed by Vens et al. (2010). Finally, in the context of genome segmentation, the more specific task of detecting isochore boundaries has been addressed (Haiminen and Mannila 2007): Simplified, isochores are large-scale structures on genomes that are visible in microscope images and correspond well (but

not perfectly) with GC rich areas of the genome. This problem has been adressed by techniques such as constrained sequence segmentation (Bingham 2010).

More information on the IQ project and its results can be found at the **project website** http://iq.ijs.si.

Acknowledgments. This work was carried out within the project IQ (Inductive Queries for Mining Patterns and Models), funded by the European Comission of the EU within FP6-IST, FET branch, under contract number 516169. Special thanks are due to the members of the project for contributing the material surveyed here.

The author is currently supported by the Slovenian Research Agency (through the research program *Knowledge Technologies* under grant P2-0103 and the research project *Data Mining for Integrative Data Analysis in Systems Biology* under grant J2-2285); the European Commission (through the FP7 project PHAGOSYS *Systems biology of phagosome formation and maturation - modulation by intracellular pathogens* under grant number HEALTH-F4-2008-223451); the Centre of Excellence for Integrated Approaches in Chemistry and Biology of Proteins (operation no. OP13.1.1.2.02.0005 financed by the European Regional Development Fund (85%) and the Slovenian Ministry of Higher Education, Science and Technology (15%)); and the Jozef Stefan International Postgraduate School.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proc. ACM SIGMOD Conf. on Management of Data, pp. 207–216. ACM Press, New York (1993)
2. Bayardo, R. (guest ed.): Constraints in data mining. Special issue of SIGKDD Explorations 4(1) (2002)
3. Becquet, C., Blachon, S., Jeudy, B., Boulicaut, J.-F., Gandrillon, O.: Strong-association-rule mining for large-scale gene-expression data analysis: a case study on human SAGE data. Genome Biology 3(12), research0067 (2002)
4. Besson, J., Boulicaut, J.-F., Guns, T., Nijssen, S.: Generalizing Itemset Mining in a Constraint Programming Setting. In: [25], pp. 107–126 (2010)
5. Bingham, E.: Finding Segmentations of Sequences. In: [25], pp. 177–197 (2010)
6. Bistarelli, S., Bonchi, F.: Interestingness is Not a Dichotomy: Introducing Softness in Constrained Pattern Mining. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 22–33. Springer, Heidelberg (2005)
7. Blachon, S., Pensa, R.G., Besson, J., Robardet, C., Boulicaut, J.-F., Gandrillon, O.: Clustering formal concepts to discover biologically relevant knowledge from gene expression data. In Silico Biology 7(4-5), 467–483 (2007)
8. Blockeel, H., Calders, T., Fromont, E., Goethals, B., Prado, A., Robardet, C.: Inductive Querying with Virtual Mining Views. In: [25], pp. 265–287 (2010b)
9. Boulicaut, J.-F., Bykowski, A., Rigotti, C.: Free-sets: a condensed representation of boolean data for the approximation of frequency queries. Data Mining and Knowledge Discovery 7(1), 5–22 (2003)
10. Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.): Constraint-Based Mining and Inductive Databases. Springer, Berlin (2005)

11. Boulicaut, J.-F., Jeudy, B.: Constraint-based data mining. In: Maimon, O., Rokach, L. (eds.) The Data Mining and Knowledge Discovery Handbook, pp. 399–416. Springer, Berlin (2005)
12. Boulicaut, J.-F., Klemettinen, M., Mannila, H.: Modeling KDD processes within the inductive database framework. In: Mohania, M., Tjoa, A.M. (eds.) DaWaK 1999. LNCS, vol. 1676, pp. 293–302. Springer, Heidelberg (1999)
13. Bringmann, B., Nijssen, S., Zimmermann, A.: From Local Patterns to Classification Models. In: [25], pp. 127–154 (2010)
14. Bringmann, B., Zimmermann, A., De Raedt, L., Nijssen, S.: Don't be afraid of simpler patterns. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 55–66. Springer, Heidelberg (2006)
15. Calders, T., Goethals, B., Prado, A.B.: Integrating pattern mining in relational databases. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 454–461. Springer, Heidelberg (2006a)
16. Calders, T., Lakshmanan, L.V.S., Ng, R.T., Paredaens, J.: Expressive power of an algebra for data mining. ACM Transactions on Database Systems 31(4), 1169–1214 (2006b)
17. Calders, T., Rigotti, C., Boulicaut, J.-F.: A survey on condensed representations for frequent sets. In: Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.) Constraint-Based Mining and Inductive Databases. LNCS (LNAI), vol. 3848, pp. 64–80. Springer, Heidelberg (2006)
18. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.-F.: Data-Peeler: Constraint-based closed pattern mining in n-ary relations. In: Proc. 8th SIAM Intl. Conf. on Data Mining, pp. 37–48. SIAM, Philadelphia (2008)
19. Cerf, L., Nhan Nguyen, B.T., Boulicaut, J.-F.: Mining Constrained Cross-Graph Cliques in Dynamic Networks. In: [25], pp. 199–228 (2010)
20. De Raedt, L.: A perspective on inductive databases. SIGKDD Explorations 4(2), 69–77 (2002a)
21. De Raedt, L.: Data mining as constraint logic programming. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS (LNAI), vol. 2408, pp. 113–125. Springer, Heidelberg (2002b)
22. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In: Proc. 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 204–212. ACM Press, New York (2008)
23. De Raedt, L., Kimmig, A., Gutmann, B., Kersting, K., Santos Costa, V., Toivonen, H.: Probabilistic Inductive Querying Using ProbLog. In: [25], pp. 229–262 (2010)
24. Džeroski, S.: Towards a general framework for data mining. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 259–300. Springer, Heidelberg (2007)
25. Džeroski, S., Goethals, B., Panov, P. (eds.): Inductive Databases and Constraint-Based Data Mining. Springer, Berlin (2010)
26. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery: An overview. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 495–515. MIT Press, Cambridge (1996)
27. Fayyad, U., Piatetsky-Shapiro, G., Uthurusamy, R.: Summary from the KDD-2003 panel – "Data Mining: The Next 10 Years". SIGKDD Explorations 5(2), 191–196 (2003)
28. Garriga, G.C., Khardon, R., De Raedt, L.: On mining closed sets in multirelational data. In: Proc. 20th Intl. Joint Conf. on Artificial Intelligence, pp. 804–809. AAAI Press, Menlo Park (2007)

29. Gionis, A., Mannila, H., Mielikainen, T., Tsaparas, P.: Assessing data mining results via swap randomization. In: Proc. 12th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 167–176. ACM Press, New York (2006)
30. Haiminen, N., Mannila, H.: Discovering isochores by least-squares optimal segmentation. Gene 394(1-2), 53–60 (2007)
31. Han, J., Lakshmanan, L.V.S., Ng, R.T.: Constraint-Based Multidimensional Data Mining. IEEE Computer 32(8), 46–50 (1999)
32. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press, Cambridge (2001)
33. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Communications of the ACM 39(11), 58–64 (1996)
34. Johnson, T., Lakshmanan, L.V., Ng, R.: The 3W model and algebra for unified data mining. In: Proc. of the Intl. Conf. on Very Large Data Bases, pp. 21–32. Morgan Kaufmann, San Francisco (2000)
35. King, R.D., Schierz, A., Clare, A., Rowland, J., Sparkes, A., Nijssen, S., Ramon, J.: Inductive Queries for a Drug Designing Robot Scientist. In: [25], pp. 425–453 (2010)
36. Kramer, S., De Raedt, L., Helma, C.: Molecular feature mining in HIV data. In: Proc. 7th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 136–143. ACM Press, New York (2001)
37. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1(3), 241–258 (1997)
38. Meo, R.: Optimization of a language for data mining. In: Proc. 18th ACM Symposium on Applied Computing, pp. 437–444. ACM Press, New York (2003)
39. Mitchell, T.M.: Generalization as search. Artificial Intelligence 18(2), 203–226 (1982)
40. Nijssen, S., De Raedt, L.: IQL: A proposal for an inductive query language. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 189–207. Springer, Heidelberg (2007)
41. Panov, P., Soldatova, L., Džeroski, S.: Representing Entities in the OntoDM Data Mining Ontology. In: [25], pp. 29–58 (2010)
42. Pečkov, A., Džeroski, S., Todorovski, L.: Multi-target polynomial regression with constraints. In: Proc. Intl. Wshp. on Constrained-Based Mining and Learning, ECML/PKDD, Warsaw, pp. 61–72 (2007)
43. Pensa, R.G., Robardet, C., Boulicaut, J.-F.: Constraint-driven co-clustering of 0/1 data. In: Basu, S., Davidson, I., Wagstaff, K. (eds.) Constrained Clustering: Advances in Algorithms, Theory and Applications, pp. 145–170. Chapman & Hall/CRC Press, Boca Raton, FL (2008)
44. Rigotti, C., Mitašiūnaitė, I., Besson, J., Meyniel, L., Boulicaut, J.-F., Gandrillon, O.: Using a Solver Over the String Pattern Domain to Analyze Gene Promoter Sequences. In: [25], pp. 407–423 (2010)
45. Slavkov, I., Džeroski, S.: Analyzing Gene Expression Data with Predictive Clustering Trees. In: [25], pp. 389–406 (2010)
46. Struyf, J., Džeroski, S.: Constrained Predictive Clustering. In: [25], pp. 155–175 (2010)
47. Vanschoren, J., Blockeel, H.: Experiment Databases. In: [25], pp. 335–361 (2010)
48. Vens, C., Schietgat, L., Struyf, J., Blockeel, H., Kocev, D., Džeroski, S.: Predicting Gene Function using Predictive Clustering Trees. In: [25], pp. 365–387 (2010)
49. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proc. 17th Intl. Conf. on Machine Learning, pp. 1103–1110. Morgan Kaufmann, San Francisco (2000)
50. Yang, Q., Wu, X.: 10 Challenging problems in data mining research. International Journal of Information Technology & Decision Making 5(4), 597–604 (2006)

What's Happening in Semantic Web
... and What FCA Could Have to Do with It

Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton, Ohio

The Semantic Web [27] is gaining momentum. Driven by over 10 years of focused project funding in the US and the EU, Semantic Web Technologies are now entering application areas in industry, academia, government, and the open Web.

The Semantic Web is based on the idea of describing the meaning – or semantics – of data on the Web using metadata – data that describes other data – in the form of ontologies, which are represented using logic-based knowledge-representation languages [26]. Central to the transfer of Semantic Web into practice is the Linked Open Data effort [7], which has already resulted in the publication, on the Web, of billions of pieces of information using ontology languages. This provides the basic data needed for establishing intelligent system applications on the Web in the tradition of Semantic Web Technologies.

Despite considerable success and progress, the field of Semantic Web Technologies still requires considerable conceptual advances in order to come to its full potential [23,24,29]. Below, we briefly list some research challenges where Formal Concept Analysis (FCA) could contribute as a method, and list some of the past FCA-related work on these issues[1].

1 Ontology Generation

Semantic Web applications require knowledge represented in the form of ontologies. However, the generation of such ontologies is a formidable modeling task which requires both ontology modeling expertise, profound domain knowledge, and an understanding of technical application requirements. Any automated or semi-automated tools which make ontology generation simpler and less costly are therefore highly desirable to have.

Automated or semi-automated generation of ontologies has been studied to a considerable extent, and some of this work uses FCA as a main component [9,10,11,21,31,50]. More recently, FCA has also found application for refining, completing, and improving ontologies [3,4,6,13,19,20,38,40,41,42,46,47,48]. Indeed, it turns out that FCA is indeed used as an ontology engineering component in recent application-driven work [5,8,16,17,30,32,37].

[1] We do not claim completeness with respect to FCA-based work in Semantic Web, and we may in particular miss early work – indeed Semantic Web as a field is not clearly defined, so it is sometimes a matter of judgement or opinion whether a paper is actually a Semantic Web paper.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 18–23, 2011.

2 Ontology Merging and Alignment

Ontology Alignment [15] refers to the process of merging two or more ontologies in order to create a larger ontology for use in applications. Tools for realizing alignments are based on a variety of techniques, and their performances differ substantially depending on the task at hand (see, e.g., the evaluation in [28]). FCA has been introduced early on as a method for ontology merging [12,18,35,43,44,49].

3 Ontology-Based Interfaces

Some work has investigated the use of FCA for interfaces, e.g. for the purpose of browsing data with underlying ontologies [11,14,45]. It seems that work related to this has, so far, only be very preliminary, so there might be further potential in this, in particular when considering the rapid expansion and rising popularity of Linked Data: To date, navigating these datasets is a tedious task, and better interfaces would have significant potential.

4 Ontology Cleansing

Usability of Linked Data is significantly hampered by the fact that it is still very *raw* data in the sense that it contains many mistakes and omissions, and cannot distinguish between different points of view [24]. In order to leverage logic-based methods mediated by ontology reasoning, data would be required to be of high modeling quality.

How to bridge this data quality gap is currently an open problem. Application of traditional methods from data mining and machine learning is limited, since the output of such methods is also, usually, prone to mistakes and errors (often measured by probabilistic confidence levels). FCA as an alternative *data clustering method* may have the potential to approach this problem, perhaps in an interactive manner akin to exploration-based completion of ontologies as performed in [40]. However, there seems to be no current work on this issue.

5 Ontology Language Development

Ontology languages currently being used are constantly being improved and revised in incremental standardization processes, e.g. through the World Wide Web Consortium (W3C). Such revisions of languages are driven both by theoretical investigations and by applicability concerns. The current situation regarding ontology languages is far from stable: While main paradigms seem to be agreed upon [26,33,36], the paradigms and their interactions are still being investigated (see, e.g., [34]) with respect to foundations and practice.

There has been some work which can be understood as dealing with the question whether FCA can be used to analyze or improve knowledge representation languages [1,2,22,25]. However, this line of investigation has not been systematically investigated yet.

Acknowledgement. The author acknowledges support by the National Science Foundation under award 1017225 *III: Small: TROn—Tractable Reasoning with Ontologies.*

References

1. Baader, F., Distel, F.: A finite basis for the set of EL-implications holding in a finite model. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 46–61. Springer, Heidelberg (2008)
2. Baader, F., Distel, F.: Exploring finite models in the description logic $\mathrm{EL}_{\mathrm{gfp}}$. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS, vol. 5548, pp. 146–161. Springer, Heidelberg (2009)
3. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Veloso, M.M. (ed.) Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, Hyderabad, India, January 6-12, pp. 230–235 (2007)
4. Baader, F., Sertkaya, B.: Applying formal concept analysis to description logics. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 261–286. Springer, Heidelberg (2004)
5. Bendaoud, R., Napoli, A., Toussaint, Y.: Formal concept analysis: A unified framework for building and refining ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 156–171. Springer, Heidelberg (2008)
6. Bendaoud, R., Napoli, A., Toussaint, Y.: A proposal for an interactive ontology design process based on formal concept analysis. In: Eschenbach, C., Grüninger, M. (eds.) Formal Ontology in Information Systems, Proceedings of the Fifth International Conference, FOIS 2008, Saarbrücken, Germany, October 31-November 3. Frontiers in Artificial Intelligence and Applications, vol. 183, pp. 311–323. IOS Press, Amsterdam (2008)
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
8. Cho, W.C., Richards, D.: Ontology construction and concept reuse with formal concept analysis for improved web document retrieval. Web Intelligence and Agent Systems 5(1), 109–126 (2007)
9. Cimiano, P.: Ontology learning from text using corpus-derived formal contexts. In: Hitzler, P., Schärfe, H. (eds.) Conceptual Structures in Practice, pp. 199–222. Chapman & Hall/CRC Press (2009)
10. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. J. Artif. Intell. Res (JAIR) 24, 305–339 (2005)
11. Cimiano, P., Hotho, A., Stumme, G., Tane, J.: Conceptual knowledge processing with formal concept analysis and ontologies. In: Eklund, P.W. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 189–207. Springer, Heidelberg (2004)
12. Curé, O.: Merging expressive ontologies using formal concept analysis. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 49–58. Springer, Heidelberg (2009)

13. Distel, F.: An approach to exploring description logic knowledge bases. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA 2010. LNCS, vol. 5986, pp. 209–224. Springer, Heidelberg (2010)
14. Ducrou, J., Eklund, P.: Faceted document navigation. In: Hitzler, P., Schärfe, H. (eds.) Conceptual Structures in Practice, pp. 225–244. Chapman & Hall/ CRC (2009)
15. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
16. Fang, K., Chang, C., Chi, Y.: Using formal concept analysis to leverage ontology-based acu-point knowledge system. In: Zhang, D. (ed.) ICMB 2008. LNCS, vol. 4901, pp. 115–121. Springer, Heidelberg (2007)
17. Fu, G., Cohn, A.G.: Utility ontology development with formal concept analysis. In: Eschenbach, C., Grüninger, M. (eds.) Formal Ontology in Information Systems, Proceedings of the Fifth International Conference, FOIS 2008, Saarbrücken, Germany, October 31-November 3. Frontiers in Artificial Intelligence and Applications, vol. 183, pp. 297–310. IOS Press, Amsterdam (2008)
18. Ganter, B., Stumme, G.: Creation and merging of ontology top-levels. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS, vol. 2746, pp. 131–145. Springer, Heidelberg (2003)
19. Hacene, M.R., Fennouh, S., Nkambou, R., Valtchev, P.: Refactoring of ontologies: Improving the design of ontological models with concept analysis. In: 22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, Arras, France, October 27-29, vol. 2, pp. 167–172. IEEE Computer Society, Los Alamitos (2010)
20. Hacene, M.R., Huchard, M., Napoli, A., Valtchev, P.: A proposal for combining formal concept analysis and description logics for mining relational data. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 51–65. Springer, Heidelberg (2007)
21. Hacene, M.R., Napoli, A., Valtchev, P., Toussaint, Y., Bendaoud, R.: Ontology learning from text using relational concept analysis. In: Proceedings of the 2008 International MCETECH Conference on e-Technologies, pp. 154–163. IEEE Computer Society, Los Alamitos (2008)
22. Hitzler, P.: Default reasoning over domains and concept hierarchies. In: Biundo, S., Frühwirth, T., Palm, G. (eds.) KI 2004. LNCS (LNAI), vol. 3238, pp. 351–365. Springer, Heidelberg (2004)
23. Hitzler, P.: Towards reasoning pragmatics. In: Janowicz, K., Raubal, M., Levashkin, S. (eds.) GeoS 2009. LNCS, vol. 5892, pp. 9–25. Springer, Heidelberg (2009)
24. Hitzler, P., van Harmelen, F.: A reasonable semantic web. Semantic Web 1(1-2), 39–44 (2010)
25. Hitzler, P., Krötzsch, M.: Querying formal contexts with answer set programs. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 260–273. Springer, Heidelberg (2006)
26. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Proposed Recommendation (September 22, 2009), http://www.w3.org/TR/owl2-primer/
27. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
28. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology alignment for linked open data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 402–417. Springer, Heidelberg (2010)

29. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked Data is Merely More Data. In: Brickley, D., Chaudhri, V.K., Halpin, H., McGuinness, D. (eds.) Linked Data Meets Artificial Intelligence, pp. 82–86. AAAI Press, Menlo Park (2010)

30. Jäschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Discovering shared conceptualizations in folksonomies. J. Web Sem. 6(1), 38–53 (2008)

31. Jia, H., Newman, J., Tianfield, H.: A new formal concept analysis based learning approach to ontology building. In: Sicilia, M.-Á., Lytras, M.D. (eds.) Metadata and Semantics, Post-proceedings of the 2nd International Conference on Metadata and Semantics Research, MTSR 2007, Corfu Island in Greece, October 1-2, pp. 433–444. Springer, Heidelberg (2007)

32. Jiang, G., Pathak, J., Chute, C.G.: Formalizing ICD coding rules using formal concept analysis. Journal of Biomedical Informatics 42(3), 504–517 (2009)

33. Kifer, M.: Rule interchange format: The framework. In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 1–11. Springer, Heidelberg (2008)

34. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Proceedings of the 20th International World Wide Web Conference, WWW 2011, Hyderabad, India, March/April (to appear, 2011)

35. Li, G. y., Liu, S. p., Zhao, Y.: Formal concept analysis based ontology merging method. In: 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), pp. 279–282. IEEE, Los Alamitos (2010)

36. Manola, F., Miller, E. (eds.): Resource Description Framework (RDF). Primer. W3C Recommendation (February 10, 2004), http://www.w3.org/TR/rdf-primer/

37. Ning, L., Guanyu, L., Li, S.: Using formal concept analysis for maritime ontology building. International Forum on Information Technology and Applications 2, 159–162 (2010)

38. Rudolph, S.: Relational Exploration – Combining Description Logics and Formal Concept Analysis for Knowledge Specification. Universitätsverlag Karlsruhe (2007)

39. Rudolph, S., Krötzsch, M., Hitzler, P.: Quo vadis, CS? – on the (non)-impact of conceptual structures on the semantic web. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 464–467. Springer, Heidelberg (2007)

40. Rudolph, S., Völker, J.: A lexico-logical approach to ontology engineering. In: Pascal Hitzler, H.S. (ed.) Conceptual Structures in Practice, pp. 225–244. Chapman & Hall/CRC (2009)

41. Rudolph, S., Völker, J., Hitzler, P.: Supporting lexical ontology learning by relational exploration. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 488–491. Springer, Heidelberg (2007)

42. Sertkaya, B.: Computing the hierarchy of conjunctions of concept names and their negations in a description logic knowledge base using formal concept analysis. In: Supplementary Proceedings of the 4th International Conference on Formal Concept Analysis, ICFCA 2006, Dresden, Germany (2006)

43. de Souza, K.X.S., Davis, J., de Medeiros Evangelista, S.R.: Aligning ontologies, evaluating concept similarities and visualizing results. Journal on Data Semantics V, 211–236 (2006)

44. Stumme, G., Maedche, A.: FCA-MERGE: Bottom-up merging of ontologies. In: Nebel, B. (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, pp. 225–234. Morgan Kaufmann, San Francisco (2001)
45. Tane, J., Cimiano, P., Hitzler, P.: Query-based multicontexts for knowledge base browsing: An evaluation. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 413–426. Springer, Heidelberg (2006)
46. Völker, J.: Learning Expressive Ontologies. Studies on the Semantic Web, vol. 002. IOS Press, Amsterdam (2009)
47. Völker, J., Rudolph, S.: Fostering web intelligence by semi-automatic OWL ontology refinement. In: Main Conference Proceedings of the 2008 IEEE /WIC/ACM International Conference on Web Intelligence, WI 2008, Sydney, NSW, Australia, December 9-12. IEEE, Los Alamitos (2008)
48. Völker, J., Rudolph, S.: Lexico-logical acquisition of OWL DL axioms. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 62–77. Springer, Heidelberg (2008)
49. Zhao, Y., Wang, X., Halang, W.: Ontology mapping based on rough formal concept analysis. In: Advanced International Conference on Telecommunications/International Conference on Internet and Web Applications and Services, p. 180 (2006)
50. Zhou, W., Liu, Z.T., Zhao, Y.: Ontology learning by clustering based on fuzzy formal concept analysis. In: Annual International of Computer Software and Applications Conference, vol. 1, pp. 204–210 (2007)

Galois Connections in Axiomatic Aggregation

Bruno Leclerc

Centre d'Analyse et de Mathématique Sociales,
École des Hautes Études en Sciences Sociales,
190 avenue de France, 75244 Paris cedex 13, France
leclerc@ehess.fr

Abstract. We investigate the relations between, on the one hand, Galois connections and the related types of maps and, on the other hand, the axiomatic Arrowian approach for the aggregation (or consensus) problem in lattices. In the latter one wants to "aggregate" n-tuples ($n \geq 2$) of elements of a lattice L into an element of this lattice representing their "consensus", subject to satisfying some desirable properties. The main axiom is a generalization of Arrow's [1] independence. The results consist in the characterization of convenient aggregation functions, and especially in impossibility ones when axioms turn to be incompatible. For the many applications of this theory in the domains of social choice or cluster analysis, see, e.g., the book of Day and McMorris [4]. Basic characterizations of Arrowian aggregation functions according to a specific typology of finite lattices are given by Monjardet [10]. They are extended to lattices of Galois maps (or polarized ones, that is maps appearing in Galois connections), then particularized to fuzzy preorders and hierarchical classifications, in Leclerc [7]. A unified presentation is given in Leclerc and Monjardet [8].
An FCA-related representation of Galois maps between two fixed lattices is given in Domenach and Leclerc [5] with the introduction of the so-called "biclosed" relations. As pointed out in the unifying paper of Ganter [6], the notion of biclosed relations is related to several others in the literature. The first part of the presentation will be devoted to Arrowian aggregation of biclosed relations.
In the second part, we present another relation between Aggregation theory and residuated/residual maps (those appearing in Residuation Theory [2]), which corresponds to "covariant" Galois connections. Chambers and Miller [3] and Leclerc and Monjardet [9] have recently pointed out that, in a significant class of atomistic lattices, an aggregation function is a meet-projection if and only if it is a residual mapping.

Keywords: Galois connection, Residuation theory, Axiomatic aggregation, Lattice, Biclosed relation.

References

1. Arrow, K.J.: Social Choice and Individual Values. Wiley, New York (1951)
2. Blyth, T.S., Janowitz, M.F.: Residuation theory. Pergamon Press, Oxford (1972)

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 24–25, 2011.

3. Chambers, C.P., Miller, A.D.: Rules for Aggregating Information. Social Choice and Welfare 36, 75–82 (2011)
4. Day, W.H.E., McMorris, F.R.: Axiomatic Consensus Theory in Group Choice and Biomathematics. SIAM, Philadelphia (2003)
5. Domenach, F., Leclerc, B.: Biclosed binary relations and Galois connections. Order 18, 89–104 (2001)
6. Ganter, B.: Relational Galois connections. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 1–17. Springer, Heidelberg (2007)
7. Leclerc, B.: Aggregation of fuzzy preferences: a theoretic Arrow-like approach. Fuzzy Sets and Systems 43, 291–309 (1991)
8. Leclerc, B., Monjardet, B.: Latticial theory of consensus. In: Barnett, V., Moulin, H., Salles, M., Schofield, N. (eds.) Social Choice, Welfare and Ethics, pp. 145–159. Cambridge University Press, Cambridge (1995)
9. Leclerc, B., Monjardet, B.: Aggregation and residuation (2011) (submitted)
10. Monjardet, B.: Arrowian characterizations of latticial federation consensus functions. Mathematical Social Sciences 20, 51–71 (1990)

Backing Composite Web Services Using Formal Concept Analysis

Zeina Azmeh[1], Fady Hamoui[1,3], Marianne Huchard[1], Nizar Messai[2], Chouki Tibermacine[1], Christelle Urtado[3], and Sylvain Vauttier[3]

[1] LIRMM - CNRS & Univ. Montpellier II, France
{azmeh,hamoui,huchard,tibermacin}@lirmm.fr
[2] MAS - Ecole Centrale Paris, France
nizar.messai@ecp.fr
[3] LGI2P - Ecole des Mines d'Alès - Nîmes, France
{Fady.Hamoui,Christelle.Urtado,Sylvain.Vauttier}@mines-ales.fr

Abstract. A Web service is a software functionality accessible through the network. Web services are intended to be composed into coarser-grained applications. Achieving a required composite functionality requires the discovery of a collection of Web services out of the enormous service space. Each service must be examined to verify its provided functionality, making the selection task neither efficient nor practical. Moreover, when a service in a composition becomes unavailable, the whole composition may become functionally broken. Therefore, an equivalent service must be retrieved to replace the broken one, thus spending more time and effort. In this paper, we propose an approach for Web service classification based on FCA, using their operations estimated similarities. The generated lattices make the identification of candidate substitutes to a given service straightforward. Thus, service compositions can be achieved more easily and with backup services, so as to easily recover the functionality of a broken service.

Keywords: Web service classification, Formal Concept Analysis (FCA), service composition, service backups.

1 Introduction

A Web service is a software functionality accessible through the network. It exposes its functionalities to the external world by an abstract interface expressed in Web Service Description Language (WSDL) [1]. A WSDL interface is an XML-based document that describes a service's available operations, parameters, data types, and access protocols.

Web services represent the building blocks for creating composite applications. When creating a composite application, each selected Web service must fulfill a part of the application's functionality. Therefore, each service's WSDL must be analyzed to verify its provided operations, and so to decide whether to select the service or not. Then, after identifying the needed services, they can be assembled together in order to meet the desired functionality of the whole composite application.

The task of finding an appropriate service to use is hard and time-consuming, because of the large number of existing Web services nowadays. This may become even harder

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 26–41, 2011.

knowing that Web services are not guaranteed to have a continuous execution. This is due to their dynamic nature, being offered by various providers, remotely accessed, and having different quality of service (QoS) levels. Therefore, an available functioning Web service may crash and become unavailable at any time, which requires finding an equivalent one to replace it, in order to maintain the application functionality.

The real challenge lies in the fact that there is a lack of WSDL management facilities, especially after the deficiency of UDDI [2], which was originally proposed as a core Web service registry standard: *"UDDI did not achieve its goal of becoming the registry for all Web Services metadata and did not become useful in a majority of Web Services interactions over the Web"* [3]. Thus, a mechanism for organizing and indexing Web services is significantly required. This leads us to our proposition for Web service classification, which is based on Formal Concept Analysis (FCA)[4].

In our proposed approach, we consider the objects to be Web services and the attributes to be the operations offered by these services. We construct Web service lattices using many-valued contexts of similarity values calculated for each pair of operations. The generated service lattices provide us with browsing and navigation capabilities. This allows the retrieval of more general to more specific sets of services [5,6]. More general sets have lesser common operations while more specific sets have more common operations. Therefore, applying FCA to Web services provides us with a retrieval mechanism, which facilitates both selection of Web services and identification of their possible substitutes. Accordingly, it helps building composite applications as well as supporting them with backup services.

The rest of the paper is organized as follows: Section 2 defines how we adapt FCA to web services. Section 3 explains our approach along with examples and formal definitions. Section 4 demonstrates a case study using real web services. Section 5 lists and discusses the related work. Finally, Section 6 concludes the paper and describes some of our perspectives.

2 An FCA-Based Approach for Web Service Classification

In our approach, we use FCA [4] in order to construct a classification of Web services. We consider that the objects are Web services and the attributes are operations. In this way, a formal context of Web services and operations becomes $\mathbb{K} = (\mathbb{W}, \mathbb{O}, I)$, where: $\mathbb{W} = \{ws_i \mid 1 \leq i \leq n_{\mathbb{W}}\ ,\ n_{\mathbb{W}} > 1\}$ is the set of Web services. We suppose that it must contain more than one Web service. Each service offers a set of one or more operations, and the union of all of the sets of operations offered by all of the services forms the total set of operations:

$$ws_i = \{op_{ij} \mid 1 \leq i \leq n_{\mathbb{W}}\ ,\ 1 \leq j \leq n_{ws_i}\}$$
$$\mathbb{O} = \bigcup_{i=1}^{i=n_{\mathbb{W}}} ws_i$$

$(ws, op) \in I$ denotes the fact that the service $ws \in \mathbb{W}$ provides the operation $op \in \mathbb{O}$ (also read as *ws has op*). Table 1 shows an example of a formal context $(\mathbb{W}, \mathbb{O}, I)$ where $\mathbb{W} = \{Calc_1, Calc_2, Calc_3\}$ and $\mathbb{O} = \{add, sub, mul, div, pow\}$.

Table 1. A formal context for $\mathbb{W} \times \mathbb{O}$

	add	*sub*	*mul*	*div*	*pow*
$Calc_1$	×	×			×
$Calc_2$	×	×		×	×
$Calc_3$	×		×	×	

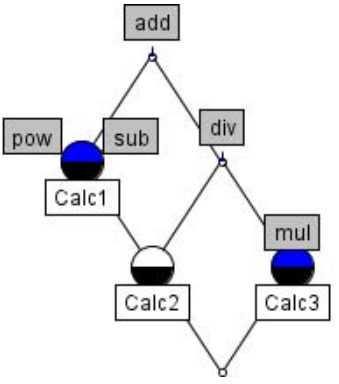


Fig. 1. The service lattice for the context in Table 1

Having a set of Web services $X \subseteq \mathbb{W}$, $X' = \{op \in \mathbb{O} \mid \forall\, ws \in X : (ws, op) \in I\}$ is the set of common operations. In the same way, having the set of operations $Y \subseteq \mathbb{O}$, $Y' = \{ws \in \mathbb{W} \mid \forall\, op \in Y : (ws, op) \in I\}$ is the set of common Web services. In our example, $(\{Calc_1, Calc_2\})' = \{add, sub, pow\}$ and $(\{div\})' = \{Calc_2, Calc_3\}$.

A concept, for example $(\{Calc_1, Calc_2\}, \{add, sub, pow\})$, is thus a maximal collection of services offering similar operations. The concept lattice defines a hierarchical organization of services and operations, in which a certain concept inherits all the extents (services) of its descendants (subconcepts) and all the intents (operations) of its ascendants (super-concepts). Fig. 1 illustrates the lattice built for the context shown in Table 1, using the *ConExp* tool [7].

From the lattice in Fig. 1, we can reveal the relationships between the presented services. We list some of them as follows:

- $Calc_1$, $Calc_2$ and $Calc_3$ offer the operation $\{add\}$. Thus, they can replace each other for this operation;
- $Calc_1$ and $Calc_2$ offer the operations $\{add, sub, pow\}$;
- $Calc_2$ can replace $Calc_1$ since it offers all of its operations in addition to *div*;
- $Calc_2$ and $Calc_3$ offer together the operations $\{add, div\}$.

Using binary contexts to classify Web services by their operations reflects two cases: the service either offers a given operation or not. Substitution can only be handled when services offer strictly identical operations which is not the case for real Web services. This is why we need to introduce the notion of operation similarity and use many-valued contexts of similarity values, as in the following section.

3 Using Many-Valued Contexts

Web services in a certain business domain may offer similar operations. In order to classify these services by their operations using FCA, we need to calculate the operation similarity and to use many-valued contexts. We explain our approach using the set of services illustrated in Table 1. For clarity, we use only the first 3 operations of each

Table 2. A set of calculation services with their operations

Services	Id	Operations	Id
$Calc_1$	ws_1	add(a,b)	op_{11}
		sub(a,b)	op_{12}
$Calc_2$	ws_2	add(a,b,c)	op_{21}
$Calc_3$	ws_3	add(a,b,c,d)	op_{31}
		sub(a,b,c)	op_{32}
		mult(a,b)	op_{33}
		add(a,b,c)	op_{34}

Table 3. The similarity matrix ($SimMat$)

	op_{11}	op_{12}	op_{21}	op_{31}	op_{32}	op_{33}	op_{34}
op_{11}	1	0	0.75	0.5	0	0	1
op_{12}	0	1	0	0	0.75	0	0
op_{21}	0.75	0	1	0.75	0	0	0.75
op_{31}	0.5	0	0.75	1	0	0	0
op_{32}	0	0.75	0	0	1	0	0
op_{33}	0	0	0	0	0	1	0
op_{34}	1	0	0.75	0	0	0	1

Table 4. The context ($SimCxt$) for $\theta = 0.75$

	op_{11}	op_{12}	op_{21}	op_{31}	op_{32}	op_{33}	op_{34}
op_{11}	×		×				×
op_{12}		×			×		
op_{21}	×		×	×			×
op_{31}			×	×			
op_{32}		×			×		
op_{33}						×	
op_{34}	×		×				×

service. We use the actual operations signatures. The set of services with their signatures are given unique identifiers, as listed in Table 2.

Next, a similarity measure must be chosen, and applied on pairs of operation signatures extracted from the WSDL files. There are several similarity measures for Web services that evaluate similarity according to syntax and semantics, such as [8,9,10]. Similarity is assessed in the form of values in the range [0,1]. If two operations are sufficiently similar, the similarity value will approach 1, otherwise it will approach 0. The similarity measure is applied on pairs of operations provided by distinct services. We do not evaluate similarity between distinct operations provided by the same service (we suppose that it is equal to 0), because when a service becomes dysfunctional, all of its operations become dysfunctional too.

A similarity measure $Sim : \mathbb{O} \times \mathbb{O} \rightarrow [0,1]$ can be defined as follows:

$\forall\, op_{ij} \in \mathbb{O} \Longrightarrow Sim(op_{ij}, op_{ij}) = 1$ (an operation with itself)
$\forall\, op_{ij}, op_{ik} \in \mathbb{O}, j \neq k \Longrightarrow Sim(op_{ij}, op_{ik}) = 0$ (operations in the same service)
$\forall\, op_{ij}, op_{nm} \in \mathbb{O}, i \neq n \Longrightarrow Sim(op_{ij}, op_{nm}) \in [0,1]$ (operations in different services)

The calculated similarity values can be presented by a symmetric square matrix that we will call $SimMat$, as shown in Table 3. This matrix is of size $n = |\mathbb{O}|$, and its diagonal elements are all equal to 1 (similarity of an operation with itself).

From the similarity matrix $SimMat$, we can extract several binary contexts, by specifying threshold values $\theta \in]0,1]$. Thus, the values of $SimMat$ that are greater or equal to the chosen threshold θ are scaled to 1, while other values are scaled to 0. The binary context that corresponds to $\theta = 0.75$ is shown in Table 4, we call it $SimCxt$.

The *SimCxt* context is a triple $(\mathbb{O}, \mathbb{O}, RSim_\theta)$, where $RSim_\theta$ is a binary relation indicating whether an operation is similar to another operation or not.

$$(op_{ij}, op_{nm}) \in RSim_\theta \iff Sim(op_{ij}, op_{nm}) \geq \theta$$

We use the *SimCxt* context to generate a lattice of operations (Fig. 2), $\underline{\mathfrak{B}}(\mathbb{O}, \mathbb{O}, RSim_\theta)$. This lattice helps in discovering groups of similar operations, which are used later on to construct the service lattice.

In the resulting operation lattice, groups of mutually similar operations can be identified by the concepts having equal extent and intent sets. We call such concepts square concepts [11], because they form square gatherings on the binary context matrix. We define a group G_{op} of mutually similar operations Op_{Sim} as:

$$G_{op} = \{Op_{Sim} \mid (Op_{Sim}, Op_{Sim}) \in \mathfrak{B}(\mathbb{O}, \mathbb{O}, RSim_\theta)\}$$

The notion of square concepts can be better recognized by performing a mutual column-line interchange in the *SimCxt*. The resulting interchanged context is shown in Table 5.

Table 5. The interchanged (*SimCxt*) context

	op_{11}	op_{34}	op_{21}	op_{31}	op_{12}	op_{32}	op_{33}
op_{11}	×	×	×				
op_{34}	×	×	×				
op_{21}	×	×	×	×			
op_{31}			×	×			
op_{12}					×	×	
op_{32}					×	×	
op_{33}							×

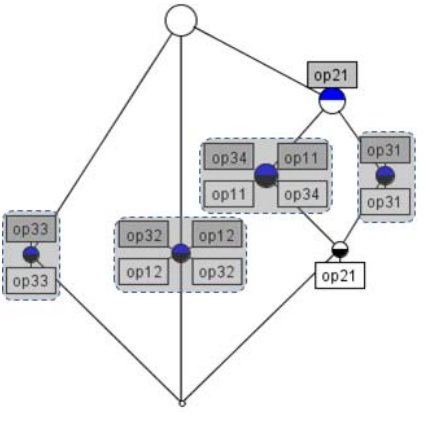


Fig. 2. The generated lattice for (*SimCxt*) shown in Table 4

From the lattice in Fig. 2 as from the interchanged context in Table 5, we can identify the groups of similar operations, and they are the following:

- $\{op_{11}, op_{34}, op_{21}\}$ that we label $(11, 34, 21)$;
- $\{op_{21}, op_{31}\}$ labelled $(21, 31)$;
- $\{op_{12}, op_{32}\}$ labelled $(12, 32)$;
- $\{op_{33}\}$ labelled (33).

The groups of similar operations, denoted as $\mathbb{G}$, are used to define the final binary context. This context is a triple $(\mathbb{W}, \mathbb{G}, R)$, in which the relation R indicates whether or not a service offers the functionality represented by the corresponding group of similar operations. We use the labels representing the groups of operations to build the final context, which is shown in Table 6. Using this context, we generate the corresponding service lattice that is shown in Fig. 3.

Table 6. The final services × groups context

	(11,34,21)	(21,31)	(12,32)	(33)
ws_1	×		×	
ws_2	×	×		
ws_3	×	×	×	×

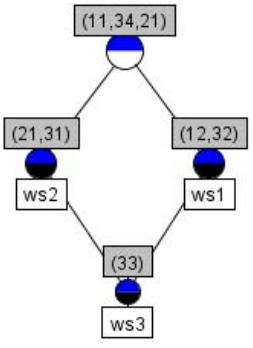


Fig. 3. The final service lattice with possible backups

From the final generated service lattice, shown in Fig. 3, we can notice the following:

- ws_1, ws_2, and ws_3 offer the functionality denoted by $(11,34,21)$, so they can replace each other for this specific functionality;
- ws_3 can replace ws_1 and ws_2, and it offers an additional functionality (33).

We can also infer immediately which services offer a specific functionality (denoted by a specific label), by considering the indices in the label. For example, the label $(11,34,21)$ makes it possible to directly deduce that (11) is provided by ws_1, (34) by ws_3 and (21) by ws_2.

4 Case Study

In this section, we demonstrate the use of service lattices for both building composite Web services and supporting them with backup services in a real world context.

We consider the example of a composite service for currency conversion, composed of two Web services: a currency converter service *Currency* and a calculation service *Calculator*. The *Currency* service offers an operation that returns the exchange rate between two entered currencies: *getRate(fromCurr,toCurr)*. The *Calculator* service offers an operation that calculates the multiplication of two entered numbers: *mul(a,b)*. We compose these two operations in order to build the composite currency service that converts a given amount from one currency to another. We describe a service composition using the Business Process Execution Language (BPEL) [12]. We use the BPEL editor of *NetBeans IDE* [13] to design and describe the specified *CompositeCurrencyService* as shown in Fig. 4.

We used the *Seekda* [14] and *Service-Finder* [15] Web service search engines to search for the needed services. We describe this case study on two parts: first we illustrate the use of the approach, then we validate it.

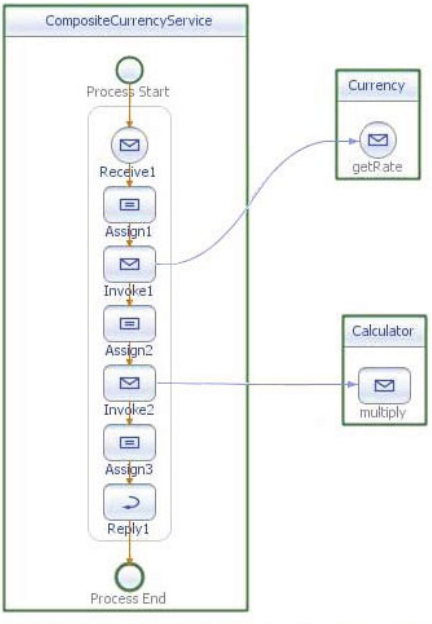


Fig. 4. The composite currency service

4.1 Using the Approach

We use a set of services for currency conversion shown in Table 7 and another set for calculation as shown in Table 8. We limit the number of services in this example, in order to simplify it and clearly explain the idea of lattice use.

For dealing with this illustration, we assess manually the similarity for the obtained services' operations of each set (an automatic approach is described later in the paper). This is achieved by comparing operation signatures (operation names, parameter names and types). Using the operations lattice and its square concepts, we identify the following groups of mutually similar operations for the currency services in Table 7:

- $\{op_{11}, op_{21}, op_{31}, op_{51}\}$ that we label $(CR : 11, 21, 31, 51)$;
- $\{op_{32}, op_{42}, op_{52}, op_{61}, op_{73}, op_{82}, op_{91}\}$ labelled $(CC : 32, 42, 52, 61, 73, 82, 91)$;
- $\{op_{33}, op_{81}\}$ labelled $(CS : 33, 81)$;
- $\{op_{41}\}$ labelled $(R : 41)$;
- $\{op_{72}\}$ labelled $(FC : 72)$;
- $\{op_{71}\}$ labelled $(CF : 71)$.

We extract also the groups of mutually similar operations for the calculation services in Table 8, and they are as follows:

Table 7. The set of currency converter services

Services	Id	Operations	Id
CurrencyConverter	ws_1	GetConversionRate(fromCurrency,toCurrency)	op_{11}
CurrencyConvertor	ws_2	ConversionRate(FromCurrency,ToCurrency)	op_{21}
DOTSCurrencyExchange	ws_3	GetExchangeRate(ConvertFromCurrency,ConvertToCurrency)	op_{31}
		ConvertCurrency(Amount,ConvertFromCurrency,ConvertToCurrency)	op_{32}
		GetCountryCurrency(Country)	op_{33}
CurrencyRates	ws_4	GetRate(CurrencyCode)	op_{41}
		GetConversion(FromCurrencyCode,ToCurrencyCode)	op_{42}
RadixxFlights	ws_5	GetExchange(FromCurrency,ToCurrency)	op_{51}
		ConvertCurrency(Amount,FromCurrency,ToCurrency)	op_{52}
rates	ws_6	Convert(CurrencyFrom,CurrencyTo,ValueFrom)	op_{61}
Conversion	ws_7	CelciusToFahrenheit(fCelsius)	op_{71}
		FahrenheitToCelcius(fFahrenheit)	op_{72}
		Currency(fValue,sFrom,sTo)	op_{73}
CurConvert	ws_8	GetCurrencySign(CountryName)	op_{81}
		ConvertCurrency(FromCountry,ToCountry,Amount)	op_{82}
ConverterService	ws_9	Convert(sourceCurrency,targetCurrency,value)	op_{91}

Table 8. The set of calculation services

Services	Id	Operations	Id
Calc	ws_1	add(a,b)	op_{11}
		div(a,b)	op_{12}
		mul(a,b)	op_{13}
		pow(b,a)	op_{14}
		sub(a,b)	op_{15}
Service	ws_2	add(a,b)	op_{21}
		sqrt(a)	op_{22}
		sub(a,b)	op_{23}
MathService	ws_3	Add(A,B)	op_{31}
		Divide(A,B)	op_{32}
		Multiply(A,B)	op_{33}
		Subtract(A,B)	op_{34}
CalculatorService	ws_4	add(y,x)	op_{41}
		divide(denominator,numerator)	op_{42}
		multiply(y,x)	op_{43}
		subtract(y,x)	op_{44}
CalcService	ws_5	Divide(A,B)	op_{51}
		Multiply(A,B)	op_{52}
		OperationAdd(A,B)	op_{53}
		Subtract(A,B)	op_{54}
Calculate	ws_6	Add(dbl1,dbl2)	op_{61}
		Divide(dbl1,dbl2)	op_{62}
		Multiply(dbl1,dbl2)	op_{63}
		Subtract(dbl1,dbl2)	op_{64}

- $\{op_{15},op_{23},op_{34},op_{44},op_{54},op_{64}\}$ labelled $(sub : 15,23,34,44,54,64)$;
- $\{op_{11},op_{21},op_{31},op_{41},op_{53},op_{61}\}$ labelled $(add : 11,21,31,41,53,61)$;
- $\{op_{13},op_{33},op_{43},op_{52},op_{63}\}$ labelled $(mul : 13,33,43,52,63)$;
- $\{op_{12},op_{32},op_{42},op_{51},op_{62}\}$ labelled $(div : 12,32,42,51,62)$;
- $\{op_{14}\}$ labelled $(pow : 14)$;
- $\{op_{22}\}$ labelled $(sqrt : 22)$.

These extracted groups of similar operations lead to a binary context for each set of services as shown in Tables 9 and 10.

We generate the two corresponding lattices as shown in the right side of Fig. 5. We can exploit these service lattices to build our composite service as well as to support it

Table 9. The formal context corresponding to the currency converter services

	(CR:11,21,31,51)	(CC:32,42,52,61,73,82,91)	(CS:33,81)	(R:41)	(FC:72)	(CF:71)
ws_1	×					
ws_2	×	×				
ws_3	×	×	×			
ws_4		×		×		
ws_5	×	×				
ws_6		×				
ws_7		×			×	×
ws_8		×	×			
ws_9		×				

Table 10. The formal context corresponding to the calculator services

	(sub:15,23,34,44,54,64)	(add:11,21,31,41,53,61)	(mul:13,33,43,52,63)	(div:12,32,42,51,62)	(pow:14)	(sqrt:22)
ws_1	×	×	×	×	×	
ws_2	×	×				×
ws_3	×	×	×	×		
ws_4	×	×	×	×		
ws_5	×	×	×	×		
ws_6	×	×	×	×		

with backup services. Thus, we decide to select operation $op_{11} : (CR : 11)$ from service ws_1 for exchange rate (currency lattice), and operation $op_{13} : (mul : 13)$ from service ws_1 for multiplication (calculation lattice). From these lattices (Fig. 5), we can also extract some backup services for our composite service according to the selected operations. For example, we used operation $op_{11} : (CR : 11)$ from service ws_1, which has 3 equivalent operations: $op_{21} : (CR : 21)$, $op_{31} : (CR : 31)$ and $op_{51} : (CR : 51)$ appearing clearly in the lattice. This means that if service ws_1 breaks down, we can replace it by any of the services ws_2 (equivalent to ws_1 being in the same concept), ws_3 or ws_5 (services introduced in subconcepts).

Moreover, if we go down in the lattice, we get the set of services that provide the operations used together with extra operations, like service ws_5 and service ws_3. They can help if the composite service evolves and needs other operations. In the same way, we can extract the backup services for the calculation service ws_1 that we are using. According to the calculation service lattice, service ws_1 as a whole set of operations cannot be replaced by any service. But, regarding the multiplication functionality, $op_{13}(mul : 13)$, it can be replaced by operations $op_{33} : (mul : 33)$, $op_{43} : (mul : 43)$, $op_{52} : (mul : 52)$, and $op_{63} : (mul : 63)$, which are offered by services ws_3, ws_4, ws_5, and ws_6 respectively. This gives us a replacement possibility in case of unavailability of ws_1 in the framework of the composite currency service.

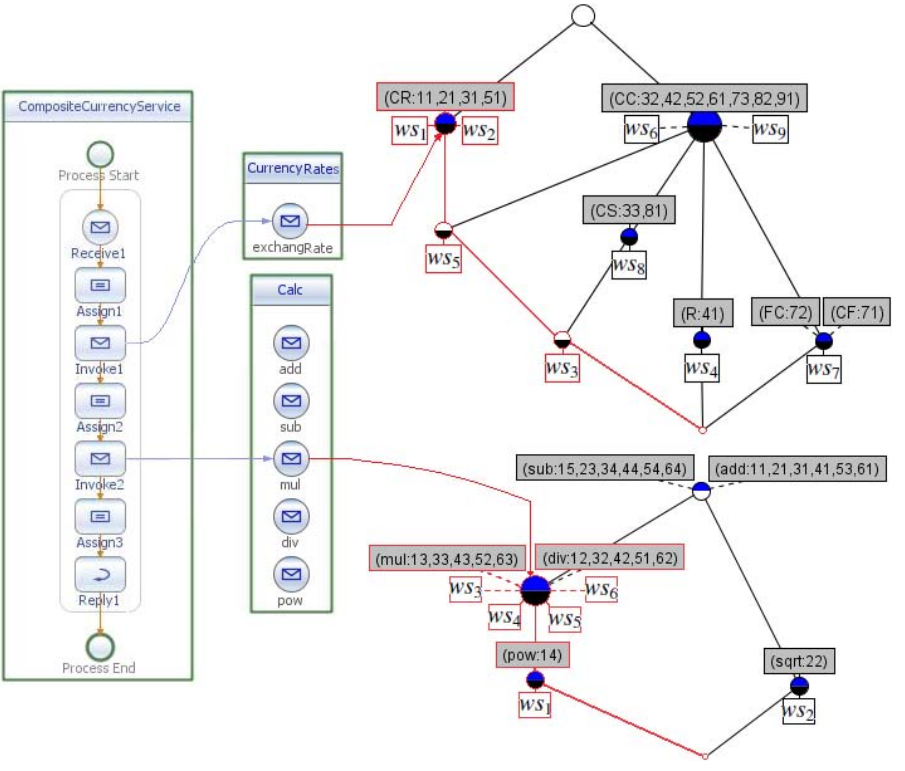


Fig. 5. The composite currency service, supported by backups from the service lattices

4.2 Validation

In this section, we validate our approach using the entire number of retrieved *Calculator* and *Currency* services[1]. We queried $Service - Finder$ to collect service endpoints (addresses), then we downloaded the corresponding WSDL interfaces via *Seekda*. For the *Calculator* service, we searched using *multiply* as keyword. This returned a set $WS1$ of 29 services, among which we found one unrelated service.

For the *Currency* service, we used a combination of the following keywords exchange, rate, currency, converter. After eliminating the repeated services, we found a set of 81 services. From this set, we also eliminated the services that we were unable to parse. This resulted in a set $WS2$ of 64 services.

We parsed each service of the two sets (WSDL parser[2]), to extract its operation signatures. The set $WS1$ has a total of 142 operations, while $WS2$ has 935 operations.

In order to calculate the *SimMat* (explained in Section 3) for both sets of services, we make use of $Jaro - Winkler$ [16] similarity measure, to assess the similarity between the extracted signatures according to each set. This metric gave convenient similarity values

[1] Retrieved services: http://www.lirmm.fr/∼azmeh/icfca11/CaseStudy.html
[2] Available online: http://www.lirmm.fr/∼azmeh/tools/WsdlParser.html

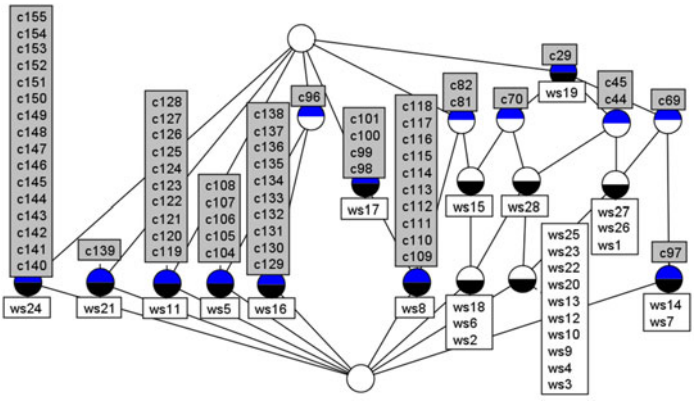


Fig. 6. The lattice corresponding to the *Calculator* services set

that were calculated efficiently, compared to another tested technique that used a combination of syntactic and semantic metrics. After a number of experiments, we found that a relatively pertinent similarity value starts from 80%. By applying this threshold on the *SimMat*, we obtained the binary *SimCxt* corresponding to each set.

We tried to compute the lattices corresponding to each *SimCxt* using Galicia [17]. The lattices could not be generated due an "out of memory" error (on a machine with limited resources). Therefore, we computed the Galois Sub Hierarchy (GSH), (order induced by attribute and object concepts). Using GSH, we obtained a suborder of 155 concepts for *SimCxt* (142×142) and another suborder of 1724 concepts for *SimCxt* (935×935). The second suborder may be reduced depending on the functionality filtering techniques.

Hereby, we restrict our analysis on $WS1$ regarding the limited paper space. From the GSH calculated for *SimCxt* (142×142), we extracted 65 square concepts. Among these 65 square concepts, we had 13 non-trivial concepts and 52 concepts reduced to one operation. Each square concept represents a functionality, for example: $c82$ represents the *multiply* functionality. It contains $\{op15.2, op18.3, op2.3, op6.3, op8.2\}$, which are mutually substitutable operations for calculating the multiplication of two numbers.

Afterwards, we constructed the lattice of services (as objects) and these square concepts (as attributes). The generated lattice is shown in Fig. 6, and contains 21 concepts. By regarding the right half of the lattice, we can notice services that can be entirely replaced by other ones. For example: if we consider $ws15$, it contains the *multiply* functionality (being a subconcept of $c82$). This service can be replaced by three other services: $ws18$, $ws6$ and $ws2$.

5 Related Work

Software engineering research has long benefitted from using FCA-based techniques in various ways, as attested by [18] which indexes and classifies 42 such scientific papers published between 1992 and 2003 using FCA. These works go from early development phases (requirement engineering) to late ones (maintenance or legacy system analysis).

Many have focused on refactoring and reingineering, especially in object-oriented languages [19,20,21,22]. Although our approach can be used and understood as a Web service refactoring method (since operations are factorized in the lattice), this paper chooses to focus on the classification of Web services inside backup service libraries. Among the works that ambition to browse or request software libraries using FCA, some rely mainly on syntax [23,24], extending type theory [25] to recent paradigms (Component-based development or SOA). Others have studied the use of FCA [26] to structure keyword-based indexes that enable to browse software libraries [27,28]. In the literature, we can find several works that more specifically focus on Web service classification and selection. A quick overview can be obtained from [29,30]. In sections 5.1 and 5.2 we list a selection of works based respectively on FCA and on other techniques. Then we discuss comparatively our contribution in Section 5.3.

5.1 Approaches Based on FCA

In [31], Web services are classified using FCA to facilitate WSDL browsing. The formal contexts are composed according to three levels, service level, operation level and type level, together with keywords. These keywords are identified from the WSDL files by applying vector space metrics with the help of WordNet to discover the synonyms. The resulting service lattice represents an indexing of Web services, it highlights the relationships between the services and permits the identification of different categorizations of a certain service. In [32], FCA is used together with keywords extracted from services' interfaces to build a Web services lattice. The extracted words are processed using WordNet and other IR techniques, and are classified into vectors using support vector machines (SVM). The obtained vectors categorize the services into domains, and service lattices are obtained for each category using FCA.

In [33,34], pairs of similar operations, depending on a chosen threshold, are merged together and the services are described by a representative operation of the pair in order to build the service lattice. They do not approach the issue that in a set of operations, op_1 can be similar to op_2 and op_2 similar to op_3 but op_1 might be not similar to op_3 because similarity is in general not a transitive operation. We solve this issue using an intermediate operation lattice based on the *SimCxt* to merge the maximal sets of mutually similar operations. Our mining of mutually similar operations is another application of the use of tolerance relations jointly with FCA, as is also done in [35].

5.2 Approaches Based on other Techniques

Many approaches use machine learning techniques, in order to discover and group similar services. In [36,37], service classifiers are defined depending on sets of previously categorized services. The resulting classifiers are then used to deduce relevant categories for new services. In case there are no predefined categories, unsupervised clustering is used. In [38], the CPLSA approach is defined that reduces a service set then clusters it into semantically related groups.

In [39], a Web service broker is designed relying on approximate signature matching using XML schema matching. It can recommend services to programmers in order to compose them. In [40], a service request and a service are represented as two finite state

machines. Then, they are compared using various heuristics to find structural similarities between them. In [8], the Woogle Web service search engine is presented, which takes the needed operation as input and searches for all the services that include an operation similar to the requested one. In [41], tags coming from folksonomies are used to discover and compose services.

The vector space model is used for service retrieval in several existing works as in [42,43,44]. Terms are extracted from every WSDL file and vectors are built for describing service. A query vector is also built, and similarity is calculated between the service vectors and the query vector. This model is sometimes enhanced by using WordNet structure matching algorithms to ameliorate similarity scores as in [43], or by partitioning the search space into smaller subspaces as in [44].

5.3 Discussion

In FCA approaches based on keywords, similar operations can not be determined and thus, Web service substitutes can not be identified either. In our approach, we generate an intermediary lattice to group mutually similar operations. Thus, sets of equivalent operations appear in each concept of the final lattice. This serves for several purposes such as service retrieval, selection and support for service compositions with backup services. Indeed, one of our main contributions is the idea of supporting the continuity of service compositions. When selecting a service, a sub-lattice that is descendant from this service can be extracted. This sub-lattice contains the possible backups that can replace this service to ensure a recovered functionality.

A service lattice is a structure that reveals relations between services according to the operations provided in common. It offers a navigation facility that enables better discovery and browsing than in structures such as lists and sets used in the other approaches. New services can be classified in existing lattices using incremental lattice generation algorithms. Thus, there is no need to regenerate the whole lattice.

Moreover, our approach can be tuned to have similarity thresholds set to consider finer-grained to coarser-grained comparisons. Indeed, the threshold values set during the process set the sizes of the sieves used to keep similar operations. These thresholds are set empirically. They condition the number of candidate backups our approach will discover. If there are too much candidate services, selection might as well be harder (only very similar services should be kept): the threshold can be raised. If there are few services proposed as backups, the threshold can be lowered. Backup candidates will be more dissimilar, probably requiring some little manual adaptations, but such setting would find backup possibilities where others would not. Finally, systematically calculating lattices for several threshold values might provide an interesting zoom-in / zoom-out capability that would provide several finer to coarser-grained classifications as in [45].

6 Conclusion and Future Work

In this paper, we proposed an approach based on Formal Concept Analysis (FCA) for building Web service lattices according to functionality domains. We make use of similarity measures for Web services to form our formal contexts in order to build the lattices according to threshold values.

A Web service lattice reveals the invisible relations between Web services in a certain domain, showing the services that are able to replace other ones. Thus, facilitating service browsing, selecting and identifying possible substitutions. We explained how to exploit the resulting lattices to build orchestrations of Web services and supporting them with backup services.

The quality of our generated lattices depends on the chosen similarity measure [8,9,10] and the similarity threshold. The more accurate the measure is, the more precise the obtained lattice is. The chosen values of threshold will give us a variation of lattices, and they reflect the level of the required adaptations. Thus, a high value of threshold means similar services with a low number of required adaptations.

Our work in progress is to enrich the service lattices with quality of service (QoS) aspects, in order to enable an automatic selection of a service that responds to a requested level of QoS. We are also working on the dynamic substitution of a Web service by one of its backups, to ensure a continuous functionality of a service orchestration. Besides, the construction of a composite web service could benefit from Relational Concept Analysis [46]. Several context families could be considered that would encode relations between operations, operations and services, or services in the composition.

Another challenge is the dynamic update of the classification. As algorithms exist that incrementally build lattices, we believe adding services might be possible without reconsidering the whole calculus. When the disappearance of services is concerned, dismissing the indexing information immediately might not be a good idea as services might be frequently unavailable for temporary periods of time (as a crashed web server reboots, for instance). More observation still is necessary for us to evaluate if disappearances should be handled as immediate removals (or, maybe, as lazy removals, based on their being unavailable for too long). This dynamic aspect is an interesting field for future research.

Acknowldegements. Authors would like to thank anonymous reviewers for their relevant comments that helped to clarify and enrich our paper.

References

1. Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl
2. UDDI Version 3.0.2, http://www.uddi.org/pubs/uddi_v3.htm
3. Newcomer, E., Lomow, G.: Understanding SOA with Web Services (Independent Technology Guides). Addison-Wesley Professional, Reading (2004)
4. Ganter, B., Wille, R.: Formal Concept Analysis. Mathematical foundations edn. Springer, Heidelberg (1999)
5. Godin, R., Mineau, G.W., Missaoui, R.: Méthodes de classification conceptuelle basées sur les treillis de Galois et applications. Revue d'intelligence Artificielle 9, 105–137 (1995)
6. Carpineto, C., Romano, G.: A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning 24, 95–122 (1996)
7. The Concept Explorer, http://conexp.sourceforge.net/
8. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: Proc. of VLDB 2004, VLDB Endowment, pp. 372–383 (2004)
9. Stroulia, E., Wang, Y.: Structural and semantic matching for assessing web-service similarity. Int. J. Cooperative Inf. Syst. 14, 407–438 (2005)

10. Kokash, N.: A comparison of web service interface similarity measures. In: Proc. of STAIRS 2006, pp. 220–231. IOS Press, Amsterdam (2006)
11. Azmeh, Z., Huchard, M., Messai, N., Tibermacine, C., Urtado, C., Vauttier, S.: Many-Valued Concept Lattices for Backing Composite Web Services. Technical Report, LIRMM (2010)
12. Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
13. NetBeans IDE, http://www.netbeans.org/
14. Seekda Web Services Search Engine, http://webservices.seekda.com
15. Service-Finder Web Services Search Engine, http://demo.service-finder.eu
16. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Kambhampati, S., Knoblock, C.A. (eds.) IIWeb, pp. 73–78 (2003)
17. Galicia, http://www.iro.umontreal.ca/~galicia/
18. Tilley, T., Cole, R., Becker, P., Eklund, P.: A survey of formal concept analysis support for software engineering activities. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 250–271. Springer, Heidelberg (2005)
19. Godin, R., Mili, H.: Building and maintaining analysis-level class hierarchies using galois lattices. In: OOPSLA, pp. 394–410 (1993)
20. Snelting, G., Tip, F.: Understanding class hierarchies using concept analysis. ACM Trans. Program. Lang. Syst. 22, 540–582 (2000)
21. Huchard, M., Dicky, H., Leblanc, H.: Galois lattice as a framework to specify building class hierarchies algorithms. ITA 34, 521–548 (2000)
22. Godin, R., Valtchev, P.: Formal concept analysis-based class hierarchy design in object-oriented software development. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 304–323. Springer, Heidelberg (2005)
23. Aboud, N.A., Arévalo, G., Falleri, J.R., Huchard, M., Tibermacine, C., Urtado, C., Vauttier, S.: Automated architectural component classification using concept lattices. In: Proc. of WICSA/ECSA 2009. IEEE Computer Society Press, Cambridge (2009)
24. Arévalo, G., Desnos, N., Huchard, M., Urtado, C., Vauttier, S.: FCA-based service classification to dynamically build efficient software component directories. International Journal of General Systems 38, 427–453 (2009)
25. Liskov, B.: Keynote address - data abstraction and hierarchy. SIGPLAN Not. 23, 17–34 (1987)
26. Lindig, C.: Concept-based component retrieval. In: Köhler, J., Giunchiglia, F., Green, C., Walther, C. (eds.) Working Notes of the IJCAI 1995 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs, Montréal, Canada, pp. 21–25 (1995)
27. Fischer, B.: Specification-based browsing of software component libraries. In: Proc. of ASE 1998, Honolulu, USA, pp. 74–83 (1998)
28. Sigonneau, B., Ridoux, O.: Indexation multiple et automatisée de composants logiciels. Technique et Science Informatiques 25, 9–42 (2006)
29. Brockmans, S., Erdmann, M., Schoch, W.: Service-finder deliverable d4.1. research report about current state of the art of matchmaking algorithms. Technical report (2008)
30. Lausen, H., Steinmetz, N.: Survey of current means to discover web services. Technical report, Semantic Technology Institute, STI (2008)
31. Aversano, L., Bruno, M., Canfora, G., Penta, M.D., Distante, D.: Using concept lattices to support service selection. Int. J. Web Service Res. 3, 32–51 (2006)
32. Bruno, M., Canfora, G., Penta, M.D., Scognamiglio, R.: An approach to support web service classification and annotation. In: EEE, pp. 138–143. IEEE Computer Society, Los Alamitos (2005)
33. Peng, D., Huang, S., Wang, X., Zhou, A.: Concept-based retrieval of alternate web services. In: Zhou, L., Ooi, B.C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 359–371. Springer, Heidelberg (2005)

34. Azmeh, Z., Huchard, M., Tibermacine, C., Urtado, C., Vauttier, S.: Using concept lattices to support web service compositions with backup services. In: Proc. of ICIW 2010, pp. 363–368. IEEE Computer Society, Los Alamitos (2010)
35. Kaytoue, M., Assaghir, Z., Napoli, A., Kuznetsov, S.O.: Embedding tolerance relations in formal concept analysis: an application in information fusion. In: Huang, J., Koudas, N., Jones, G., Wu, X., Collins-Thompson, K., An, A. (eds.) CIKM, pp. 1689–1692. ACM, New York (2010)
36. Crasso, M., Zunino, A., Campo, M.: Awsc: An approach to web service classification based on machine learning techniques. Inteligencia Artificial, Revista Iberoamericana de Interligencia Artificial 12(37), 25–36 (2008)
37. Heß, A., Kushmerick, N.: Learning to attach semantic metadata to web services. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 258–273. Springer, Heidelberg (2003)
38. Ma, J., Zhang, Y., He, J.: Efficiently finding web services using a clustering semantic approach. In: Proc. of CSSSIA 2008, pp. 1–8. ACM, New York (2008)
39. Lu, J., Yu, Y.: Web service search: Who, when, what, and how. In: WISE Workshops, pp. 284–295 (2007)
40. Günay, A., Yolum, P.: Structural and semantic similarity metrics for web service matchmaking. In: Psaila, G., Wagner, R. (eds.) EC-Web 2007. LNCS, vol. 4655, pp. 129–138. Springer, Heidelberg (2007)
41. Bouillet, E., Feblowitz, M., Feng, H., Liu, Z., Ranganathan, A., Riabov, A.: A folksonomy-based model of web services for discovery and automatic composition. In: IEEE International Conference on Services Computing (SCC), pp. 389–396. IEEE Computer Society, Los Alamitos (2008)
42. Platzer, C., Dustdar, S.: A vector space search engine for web services. In: Third IEEE European Conference on Web Services, ECOWS 2005, pp. 62–71 (2005)
43. Wang, Y., Stroulia, E.: Semantic structure matching for assessing web service similarity. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 194–207. Springer, Heidelberg (2003)
44. Crasso, M., Zunino, A., Campo, M.: Query by example for web services. In: Proc. of SAC 2008, pp. 2376–2380. ACM, New York (2008)
45. Messai, N., Devignes, M.D., Napoli, A., Smaïl-Tabbone, M.: Using domain knowledge to guide lattice-based complex data exploration. In: Proc. of ECAI 2010, pp. 847–852. IOS Press, Amsterdam (2010)
46. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. 49, 39–76 (2007)

Enumerating Minimal Hypotheses and Dualizing Monotone Boolean Functions on Lattices

Mikhail A. Babin and Sergei O. Kuznetsov

State University Higher School of Economics,
School for Applied Mathematics and Information Science Pokrovskii bd. 11, 109028
Moscow, Russia
mikleb@yandex.ru, skuznetsov@hse.ru

Abstract. Any monotone Boolean function on a lattice can be described by the set of its minimal 1 values. If a lattice is given as a concept lattice, this set can be represented by the set of minimal hypotheses of a classification context. Enumeration of minimal hypotheses in output polynomial time is shown to be impossible unless P = NP, which shows that dualization of monotone functions on lattices with quasipolynomial delay is hardly possible.

1 Introduction

One of the first models of machine learning that used lattices (closure systems) was the JSM-method[1] of automated hypothesis generation [1,2]. In this model positive hypotheses are sought among intersections of positive example descriptions (object intents), same for negative hypotheses. For classification purposes, it suffices to have hypotheses minimal by inclusion, so-called minimal hypotheses. It is well-known that hypotheses may be generated with polynomial delay [7]. However, the problem of generating minimal hypotheses with polynomial delay remained an open one. In our paper we prove that minimal hypotheses cannot be generated with polynomial delay unless P=NP. This finding has an important implication for the theory of monotone Boolean functions.

The rest of the paper is organized as follows: In the second section we give most important definitions, in the third section we prove the main result about minimal hypotheses, and in the fourth section we discuss the implication of this result for the problem of dualizing monotone Boolean functions.

2 Main Definitions

We use standard definitions from [4]. Let G and M be sets, called the set of objects and attributes, respectively. Let I be a relation $I \subseteq G \times M$ between objects and attributes: for $g \in G, m \in M, gIm$ holds iff the object g has the

[1] Called so in honor of the English philosopher John Stuart Mill, who introduced methods of inductive reasoning in 19th century.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 42–48, 2011.

attribute m. The triple $\mathbb{K} = (G, M, I)$ is called a *(formal) context*. If $A \subseteq G, B \subseteq M$ are arbitrary subsets, then the *Galois connection* is given by the following *derivation operators*:

$$A' = \{m \in M \mid gIm \ \forall g \in A\}$$

$$B' = \{g \in G \mid gIm \ \forall m \in B\}$$

The pair (A, B), where $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$ is called a *(formal) concept (of the context $\mathbb{K}$)* with *extent* A and *intent* B (in this case we have also $A'' = A$ and $B'' = B$). The set of attributes B is *implied by the set of attributes* A, or implication $A \to B$ holds, if all objects from G that have all attributes from the set A also have all attributes from the set B, i.e. $A' \subseteq B'$.

Now we present a learning model from [1,2] in terms of FCA [8]. This model complies with the common paradigm of learning from positive and negative examples (see, e.g. [8], [7]): given a positive and negative examples of a "target attribute", construct a generalization of the positive examples that would not cover any negative example.

Assume that w is a target (*functional*) attribute, different from attributes from the set M, which correspond to *structural* attributes of objects. For example, in pharmacological applications the structural attributes can correspond to particular subgraphs of molecular graphs of chemical compounds.

Input data for learning can be represented by sets of positive, negative, and undetermined examples. *Positive examples* (or (+)-examples) are objects that are known to have the attribute w and *negative examples* (or (−)-examples) are objects that are known not have this attribute.

Definition 1. *Consider positive context $\mathbb{K}_+ = (G_+, M, \mathcal{I}_+)$ and negative context $\mathbb{K}_- = (G_-, M, \mathcal{I}_-)$. The context $\mathbb{K}_\pm = (G_+ \cup G_-, M \cup \{w\}, \mathcal{I}_+ \cup \mathcal{I}_- \cup G_+ \times \{w\})$ is called a* learning context*. The derivation operator in this context is denoted by superscript $\pm$.*

Definition 2. *The subset $H \subseteq M$ is called a positive (or (+)-)-hypothesis of learning context $\mathbb{K}_\pm$ if H is intent of $\mathbb{K}_+$ and H is not a subset of any intent of $\mathbb{K}_-$.*

In the same way negative (or (-)-) hypotheses are defined.

Besides classified objects (positive and negative examples), one usually has objects for which the value of the target attribute is unknown. These examples are usually called undetermined examples, they can be given by a context $\mathbb{K}_\tau := (G_\tau, M, I_\tau)$, where the corresponding derivation operator is denoted by $(\cdot)^\tau$.

Hypotheses can be used to classify the undetermined examples: If the intent

$$g^\tau := \{m \in M \mid (g, m) \in I_\tau\}$$

of an object $g \in G_\tau$ contains a positive, but no negative hypothesis, then g^τ is *classified positively*. Negative classifications are defined similarly. If g^τ contains

hypotheses of both kinds, or if g^τ contains no hypothesis at all, then the classification is contradictory or undetermined, respectively. In this case one can apply probabilistic techniques.

In [6], [7] we argued that one can restrict to *minimal* (w.r.t. inclusion $\subseteq$) hypotheses, positive as well as negative, since an object intent obviously contains a positive hypothesis if and only if it contains a minimal positive hypothesis.

Definition 3. *Let $G = \{g_1, \ldots, g_n\}$ and $M = \{m_1, \ldots, m_n\}$ be sets with same cardinality. Then the context $\mathbb{K} = (G, M, \mathcal{I}_{\neq})$ is called* contranominal scale, *where $\mathcal{I}_{\neq} = G \times M - \{(g_1, m_1), \ldots, (g_n, m_n)\}$.*

The contranominal scale has the following property, which we will use later: for any $H \subseteq M$ one has $H'' = H$ and $H' = \{g_i \mid m_i \notin H, 1 \leq i \leq n\}$.

3 Enumeration of Minimal Hypotheses

Here we discuss algorithmic complexity of enumerating all minimal hypotheses. Note that there is an obvious algorithm for enumerating all hypotheses (not necessary minimal) with polynomial delay [7]. This algorithm is an adaptation of an algorithm for computing the set of all concepts, where the branching condition is changed.

Problem: Minimal hypotheses enumeration (MHE)
INPUT: Positive and negative contexts $\mathbb{K}_+ = (G_+, M, \mathcal{I}_+), \mathbb{K}_- = (G_-, M, \mathcal{I}_-)$
OUTPUT: All minimal hypotheses of $\mathbb{K}_\pm$.

Unfortunately, this problem cannot be solved in output polynomial time unless $P = NP$. In order to prove this result we study complexity of the following decision problem.

Problem: Additional minimal hypothesis (AMH)
INPUT: Positive and negative contexts $\mathbb{K}_+ = (G_+, M, \mathcal{I}_+), \mathbb{K}_- = (G_-, M, \mathcal{I}_-)$ and a set of minimal hypotheses $\mathcal{H} = \{H_1, \ldots, H_k\}$.
QUESTION: Is there an *additional* minimal hypothesis H of $\mathbb{K}_\pm$ i.e. minimal hypothesis H that is $H \notin \mathcal{H}$.

We reduce the most known NP-complete problem satisfiability of CNF to AMH.

Problem: CNF satisfability (SAT)
INPUT: A Boolean CNF formula $f(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_k$
QUESTION: Is f satisfable?

Consider an arbitrary CNF instance $C_1, \ldots, C_k$ with variables $x_1, \ldots, x_n$, where $C_i = (li_1 \vee \ldots \vee l_{ir_i}), 1 \leq i \leq k$ and $l_{ij} \in \{x_1, \ldots, x_n\} \cup \{\neg x_1, \ldots, \neg x_n\}$ $(1 \leq i \leq k$, $1 \leq j \leq r_i)$ are some variables or their negations called literals. From this instance we construct a positive context $\mathbb{K}_+ = (G_+, M, \mathcal{I}_+)$ and a negative context

$\mathbb{K}_- = (G_-, M, \mathcal{I}_-)$. Define

$$M = \{C_1, \ldots, C_k\} \cup \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$$

$$G_+ = \{g_{x_1}, g_{\neg x_1}, \ldots, g_{x_n}, g_{\neg x_n}\} \cup \{g_{C_1}, \ldots, g_{C_k}\}$$

$$G_- = \{g_{l_1}, \ldots, g_{l_n}\}$$

The incidence relation of the positive context is defined by $\mathcal{I}_+ = \mathcal{I}_\mathcal{C} \cup \mathcal{I}_{\neq} \cup \mathcal{I}_=$, where

$$\begin{aligned}\mathcal{I}_\mathcal{C} = \{&(g_{x_i}, C_j) \mid x_i \notin C_j, 1 \leq i \leq n, 1 \leq j \leq k\} \\ &\cup \{(g_{\neg x_i}, C_j) \mid \neg x_i \notin C_j, 1 \leq i \leq n, 1 \leq j \leq k\}\end{aligned}$$

$$\begin{aligned}\mathcal{I}_{\neq} = &\{g_{x_1}, g_{\neg x_1}, \ldots, g_{x_n}, g_{\neg x_n}\} \times \{x_1, \neg x_1, \ldots, x_n, \neg x_n\} \\ &- \{(g_{x_1}, x_1), (g_{\neg x_1}, \neg x_1), \ldots, (g_{x_n}, x_n), (g_{\neg x_n}, \neg x_n)\}\end{aligned}$$

$$\mathcal{I}_= = \{(g_{C_1}, C_1), \ldots, (g_{C_k}, C_k)\}$$

that is for i-th clause $C_i^+ \cap \{g_{x_1}, g_{\neg x_1}, \ldots, g_{x_n}, g_{\neg x_n}\}$ is the set of literals not included in C_i, $\mathcal{I}_{\neq}$ is relation of contranominal scale.

The incidence relation of the negative context is given by $\mathcal{I}_- = \mathcal{I}_\mathcal{L}$ where

$$\begin{aligned}\mathcal{I}_\mathcal{L} = &G_- \times \{x_1, \neg x_1, \ldots, x_n, \neg x_n\} \\ &- \{(g_{l_1}, x_1), (g_{l_1}, \neg x_1), \ldots, (g_{l_n}, x_n), (g_{l_n}, \neg x_n)\}.\end{aligned}$$

		$C_1\ C_2\ \cdots\ C_k$	$x_1\ \neg x_1\ \cdots\ x_n\ \neg x_n$
$\mathbb{K}_+$	g_{x_1} $g_{\neg x_1}$ $\vdots$ g_{x_n} $g_{\neg x_n}$	$\mathcal{I}_\mathcal{C}$	$\mathcal{I}_{\neq}$
	g_{C_1} $\vdots$ g_{C_k}	$\mathcal{I}_=$	
$\mathbb{K}_-$	g_{l_1} $\vdots$ g_{l_n}		$\mathcal{I}_\mathcal{L}$

As the set of minimal hypotheses we take $\mathcal{H} = \{\{C_1\}, \{C_2\}, \ldots, \{C_k\}\}$. It is easy to see that $\mathbb{K}_\pm$ with $\mathcal{H}$ is a correct instance of AMH.

If a hypothesis (not necessary minimal) is not included in $\mathcal{H}$ we will call it *additional*.

Proposition 4. *If H is an additional minimal hypothesis of $\mathbb{K}_\pm$ then $H \subseteq \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$.*

Proof. Suppose $H \not\subseteq \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ then since H is not empty there is some $C_i \in H$, $1 \leq i \leq k$. But H is a minimal hypothesis and thus it does not contain any hypothesis. Hence $H = C_i$ and this contradicts that H is an *additional* minimal hypothesis. □

For any $H \subseteq \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ that for an $1 \leq i \leq n$ satisfies $\{x_i, \neg x_i\} \not\subseteq H$ we define the truth assignment ϕ_H in a natural way:

$$\phi_H(x_i) = \begin{cases} true, & \text{if } x_i \in H; \\ false, & \text{if } x_i \notin H; \end{cases}$$

In the case $\{x_i, \neg x_i\} \subseteq H$ for some $1 \leq i \leq n$, ϕ_H is not defined.

Symmetrically, for a truth assignment ϕ define the set $H_\phi = \{x_i \mid \phi(x_i) = true\} \cup \{\neg x_i \mid \phi(x_i) = false\}$.

Below, for the sake of convenience, if $H \subseteq \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ we will denote the complement of H in $\{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ by $\overline{H}$.

Proposition 5. *If a subset $H \subseteq \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ is not contained in the intent of any negative concept (i.e $\forall g \in G_-, H \not\subseteq g^-$), then $\phi_{\overline{H}}$ is correctly defined. Conversely, for a truth assignment ϕ the set $\overline{H_\phi}$ is not contained in the intent of any negative concept.*

Proof. The proof is straightforward. □

The following theorem proves NP-hardness of AMH.

Theorem 6. *AMH has a solution if and only if SAT has a solution.*

Proof.($\Rightarrow$) Let H be an additional minimal hypothesis of $\mathbb{K}_\pm$. First note that by Proposition 4 and Proposition 5 the truth assignment $\phi_{\overline{H}}$ is correctly defined. Since H is a nonempty concept intent of $\mathbb{K}_+$, Proposition 4 together with the fact that $I_{\neq}$ is the relation of contranominal scale implies $H^+ = \{g_{x_i} \mid x_i \in \overline{H}\} \cup \{g_{\neg x_i} \mid \neg x_i \in \overline{H}\}$. Now $H^{++} \cap \{C_1, C_2, \ldots, C_k\} = \emptyset$, hence for any C_i $(1 \leq i \leq k)$ there is some $g_l \in H^+$ such that $g_l \notin C_i^+$. According to the definition of $\mathcal{I}_\mathcal{C}$ the letter means that literal l belongs to clause C_i. Thus $f(\phi_{\overline{H}}) = true$.

($\Leftarrow$) Let ϕ be a truth assignment and $f(\phi) = true$. Define $H = \overline{H_\phi}$. Note that $H^+ = \{g_{x_i} \mid x_i \in H_\phi\} \cup \{g_{\neg x_i} \mid \neg x_i \in H_\phi\}$, because $\mathcal{I}_{\neq}$ is the relation of contranominal scale and $H \cap g^+_{C_j} = \emptyset, 1 \leq i \leq k$. Suppose that $C_i \in H^{++}$ for some $1 \leq i \leq k$. This is equivalent to $H^+ \subseteq C_i^+$. Hence, by definition of $\mathcal{I}_\mathcal{C}$, there is no literal $l \in H_\phi$ such that $l \in C_i$. Therefore, the clause C_i does not hold and this contradicts that ϕ satisfies CNF f. Thus $H^{++} = H$ and H is a hypothesis. Since H does not contain any $\{C_i\}$, it must contain additional minimal hypothesis. □

Corollary 1. *MHE cannot be solved in output polynomial time, unless $P = NP$.*

Proof. Assume there is an output polynomial algorithm $\mathcal{A}$ that generates all

minimal hypotheses in time $pol(|G_+|, |M|, |\mathcal{I}_+|, |G_-|, |\mathcal{I}_-|, N)$, where N is the number of minimal hypotheses. Use this algorithm to construct $\mathcal{A}'$ that makes first $p(|G_+|, |M|, |\mathcal{I}_+|, |G_-|, |\mathcal{I}_-|, k+1)$ steps of $\mathcal{A}$. Clearly, if there is more than k minimal hypotheses, then $\mathcal{A}'$ generates $k+1$ minimal hypotheses, hence we can solve AMH in polynomial time. □

4 Dualizing Monotone Boolean Functions on Lattices

Let $\mathfrak{B}$ be a complete lattice and f be a monotone Boolean function on it. Without loss of generality we can assume that $\mathfrak{B}$ is a concept lattice $\mathfrak{B}(G, M, I)$ from the corresponding formal context $\mathbb{K}(G, M, I)$. Then $A \subseteq B \Rightarrow f((A, A')) \leq f((B, B'))$. It is known that any monotone Boolean function on a lattice is uniquely given by its minimal 1 values, i.e. by the set $\{(A, A') \mid (A, A') \in \mathfrak{B}, f((A, A')) = 1, f((B, B')) = 0 \; \forall B \subset A\}$. We can represent the set of minimal 1 values of a monotone Boolean function as the set of minimal hypotheses of the learning context defined by $\mathbb{K}_+$ and $\mathbb{K}_-$, where $\mathbb{K}_+ = \mathbb{K}$ and object intents of $\mathbb{K}_-$ are precisely maximal 0 values of f. Symmetrically, a learning context $\mathbb{K}_\pm$ specifies a monotone Boolean function f on concept lattice of $\mathbb{K}_+$ such that maximal 0 values of f are (inclusion) maximal object intents of $\mathbb{K}_-$. Consider the following

Problem: Minimal true values enumeration (MTE)
INPUT: A formal context $\mathbb{K}$ and a set of maximal 0 values of monotone Boolean function f on the concept lattice of $\mathbb{K}$
OUTPUT: Set of minimal 1 values of f.

From Corollary 1 it follows that MTE cannot be solved in output polynomial time unless $P = NP$. Note that in the case of Boolean lattice this problem is polynomially equivalent to Monotone Boolean Dualism (see [3]) and the minimal hypotheses in this case can be enumerated with quasi-polynomial delay $O(n^{o(\log n)})$, where n is the input size.

5 Conclusions

The enumeration of minimal hypotheses of a learning context in output polynomial time was shown to be impossible unless $P = NP$. This implies that dualizing monotone Boolean functions on lattices given by their contexts is not possible in quasi-polynomial time unless $P = NP$.

Acknowledgments

The authors were supported by the project of the Russian Foundation for Basic Research, grant no. 08-07-92497-NTsNIL_a.

References

1. Finn, V.K.: On Machine-Oriented Formalization of Plausible Reasoning in the Style of F. Backon–J. S. Mill. Semiotika Informatika 20, 35–101 (1983) (in Russian)
2. Finn, V.K.: Plausible Reasoning in Systems of JSM Type. Itogi Nauki i Tekhniki, Seriya Informatika 15, 54–101 (1991) (in Russian)
3. Fredman, M.L., Khachiyan, L.: On the complexity of dualization of monotone disjunctive normal forms. Journal of Algorithms 21, 618–628 (1996)
4. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
5. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
6. Ganter, B., Kuznetsov, S.: Formalizing Hypotheses with Concepts. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 342–356. Springer, Heidelberg (2000)
7. Kuznetsov, S.O.: Complexity of Learning in Concept Lattices from Positive and Negative Examples. Discrete Applied Mathematics (142), 111–125 (2004)
8. Ganter, B., Kuznetsov, S.O.: Hypotheses and Version Spaces. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS (LNAI), vol. 2746, pp. 83–95. Springer, Heidelberg (2003)

Border Algorithms for Computing Hasse Diagrams of Arbitrary Lattices*

José L. Balcázar and Cristina Tîrnăucă

Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria
Santander, Spain
{joseluis.balcazar,cristina.tirnauca}@unican.es

Abstract. The Border algorithm and the iPred algorithm find the Hasse diagrams of FCA lattices. We show that they can be generalized to arbitrary lattices. In the case of iPred, this requires the identification of a join-semilattice homomorphism into a distributive lattice.

Keywords: Lattices, Hasse diagrams, border algorithms.

1 Introduction

Lattices are mathematical structures with many applications in computer science; among these, we are interested in fields like data mining, machine learning, or knowledge discovery in databases. One well-established use of lattice theory is in formal concept analysis (FCA) [8], where the concept lattice with its diagram graph allows the visualization and summarization of data in a more concise representation. In the Data Mining community, the same mathematical notions (often under additional "frequency" constraints that bound from below the size of the support set) are studied under the banner of Closed-Set Mining (see e.g. [21]).

In these applications, each dataset consists of *transactions*, also called *objects*, each of which, besides having received a unique identifier, consists of a set of *items* or *attributes* taken from a previously agreed finite set. A *concept* is a pair formed by a set of transactions —the *extent* set or *support set* of the concept— and a set of attributes —the *intent* set of the concept— defined as the set of all those attributes that are shared by all the transactions present in the extent. Some data analysis processes are based on the family of all intents (the "closures" stemming from the dataset); but others require to determine also their order relation, which is a finite lattice, in the form of a line graph (the *Hasse diagram*).

Existing algorithms can be divided into three main types: the ones that only generate the set of concepts, the ones that first generate the set of concepts

* This work has been partially supported by project FORMALISM (TIN2007-66523) of Programa Nacional de Investigación, Ministerio de Ciencia e Innovación (MICINN), Spain, by the Juan de la Cierva contract JCI-2009-04626 of the same ministry, and by the Pascal-2 Network of the European Union.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 49–64, 2011.

and then construct the Hasse diagram, and the ones that construct the diagram while computing the lattice elements (see [21], and also [9,12] and the references therein). The goal is to obtain the concept lattice in linear time in the number of concepts because this number is, most of the times, already exponential in the number of attributes, making the task of getting polynomial algorithms in the size of the input rather impossible.

One widespread use of concepts or closures is the generation of implications or of partial implications (also called association rules). Several data mining algorithms aim at processing large datasets in time linear in the size of the closure space, and explore closed sets individually; these solutions tend to drown the user under a deluge of partial implications. More sophisticated works attempt at providing selected "bases" of partial implications; the early proposal in [13] requires to compute immediate predecessors, that is, the Hasse diagram. Alternative proposals such as the Essential Rules of [1] or the equivalent Representative Rules of [11] (of which a detailed discussion with new characterizations and an alternative basis proposal appears in [6]) require to process predecessors of closed sets obeying tightly certain support inequalities; these algorithms also benefit from the Hasse diagram, as the slow alternatives are blind repeated traversal of the closed sets in time quadratic in the size of the closure space, or storage of all predecessors of each closed set, which soon becomes large enough to impose a considerable penalty on the running times.

The problem of constructing the Hasse diagram of an arbitrary finite lattice is less studied. One algorithm that has a better worst case complexity than various previous works is described in [16]. From our "arbitrary lattices" perspective, its main drawback is that it requires the availability of a *basis* from which each element of the lattice can be derived. In the absence of such a subset, one may still use this algorithm (at a greater computational cost) to output the Dedekind-MacNeille completion [7] of the given lattice, which in our case is isomorphic to the lattice itself. The algorithm is also easily adaptable to concept lattices, where indeed a basis is available immediately from the dataset transactions.

We consider of interest to have available further, faster algorithms for arbitrary finite lattices; we have two reasons for this aim. First, many (although not all) algorithms constructing Hasse diagrams traverse concepts in layers defined by the size of the intents; our explorations about association rules sometimes require to follow different orderings, so that a more abstract approach is helpful; second, we keep in mind the application area corresponding to certain variants of implications and database dependencies that are characterized by lattices of equivalence relations, so that we are interested in laying a strong foundation that gives us a clear picture of the applicability requirements for each algorithm constructing Hasse diagrams in lattices other than powerset sublattices.

Of course, we expect that FCA-oriented algorithms could be a good source of inspiration for the design of algorithms applicable in the general case. An example that such an extension can be done is the algorithm in [20] (see Section 3 for more details), whose highest-level description matches the general case of arbitrary lattices; nevertheless, the actual implementation described in [20]

works strictly for formal concept lattices, so that further implementations and complexity analyses are not readily available for arbitrary finite lattices.

The contribution of the present paper supports the same idea: we show how two existing algorithms that build the Hasse diagrams of a concept lattice can be adapted to work for arbitrary lattices. Both algorithms have in common the notion of *border*, which we (re-)define and formalize in Section 3, after presenting some preliminary notions about lattice theory in Section 2; our approach has the specific interest that the notion of border is given just in terms of the ordering relation, and not in terms of a set of elements already processed as in previous references ([5,14,20]); yet, the notions are equivalent. We state and prove properties of borders and describe the *Generalized Border Algorithm*; whereas the algorithm reads, in high level, exactly as in previous references, its validation is new, as previous ones depended on the lattice being an FCA lattice. In Section 4 we introduce the *Generalized iPred Algorithm*, exporting the iPred algorithm of FCA lattices [5] to arbitrary lattices, after arguing its correctness. This task is far from trivial and is our major contribution, since the existing rendering and validation of the iPred algorithm relies again extensively on the fact that it is being applied to an FCA lattice, and even performs operations on difference sets that may not belong to the closure space. Concluding remarks and future work ideas are presented in Section 5.

2 Preliminaries

We develop all our work in terms of lattices and semilattices; see [7] as main source. All our structures are *finite*. A *lattice* is a partially ordered set in which every nonempty subset has a meet (greatest lower bound) and a join (lowest upper bound). If only one of these two operations is guaranteed to be available a priori, we speak of a *join-semilattice* or a *meet-semilattice* as convenient. Top and bottom elements are denoted $\top$ and $\bot$, respectively. Lower case letters, possibly with primes, and taken usually from the end of the latin alphabet denote lattice elements: x, y'. Note that Galois connections are not explicitly present in this paper, so that the "prime" notation does not refer to the operations of Galois connections.

Finite semilattices can be extended into lattices by addition of at most one further element [7]; for instance, if $(\mathcal{L}, \leq, \vee)$ is a join-semilattice with bottom element $\bot$, one can define a meet operation as follows: $\bigwedge X = \bigvee \{y \mid \forall x \in X, y \leq x\}$; the element $\bot$ ensures that this set is nonempty. Thus, if the join-semilattice lacks a bottom element, it suffices to add an "artificial" one to obtain a lattice. A dual process is obviously possible in meet-semilattices.

Given two join-semilattices $(S, \vee)$ and $(T, \vee)$, a *homomorphism* is a function $f : S \to T$ such that $f(x \vee y) = f(x) \vee f(y)$. Hence f is just a homomorphism of the two semigroups associated with the two semilattices. If S and T both include a bottom element $\bot$, then f should also be a monoid homomorphism, i.e. we additionally require that $f(\bot) = \bot$. Homomorphisms of meet-semilattices and of lattices are defined similarly. It is easy to check that $x \leq y \Rightarrow f(x) \leq f(y)$ for

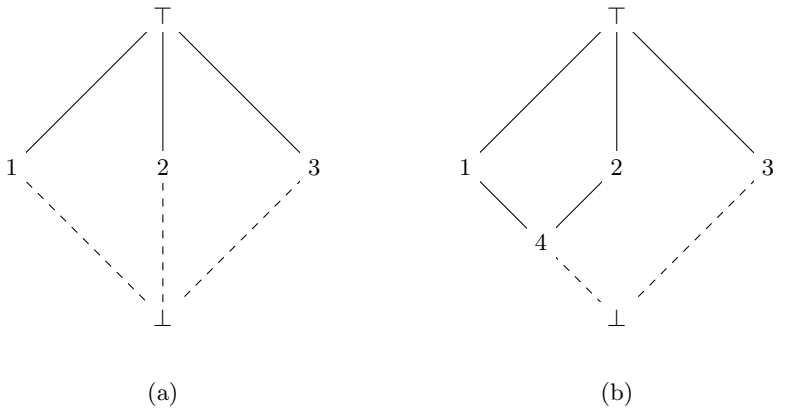


Fig. 1. Two join-semilattices converted into lattices

any homomorphism f; the converse implication, thus the equivalence $x \leq y \Leftrightarrow f(x) \leq f(y)$, is also true for injective f but not guaranteed in general.

We must point out here a simple but crucial fact that plays a role in our later developments: given a homomorphism f between two join-semilattices S and T, if we extend both into lattices as just indicated, then f is *not* necessarily a lattice homomorphism; for instance, there could be elements of T that do not belong to the image set of f, and they may become meets of subsets of T in a way that prevents them to be the image of the corresponding meet of S. For one specific example, see Figure 1: consider the two join-semilattices defined by the solid lines, where the numbering defines an injective homomorphism from the join-semilattice in (a) to the join-semilattice in (b). Both lack a bottom element. Upon adding it, as indicated by the broken lines, in lattice (a) the meets of 1 and 2 and of 1 and 3 coincide, but the meets of their corresponding images in (b) do not; for this reason, the homomorphism cannot be extended to the whole lattices.

However, the following does hold:

Lemma 1. *Consider two join-semilattices S and T, and let $f : S \to T$ be a homomorphism. After extending both semilattices into lattices, $f(\bigwedge Y) \leq \bigwedge f(Y)$ for all $Y \subseteq S$.*

This is immediate to see by considering that $\bigwedge Y \leq y$ for all $y \in Y$, hence $f(\bigwedge Y) \leq f(y)$ for all such y, and the claimed inequality follows.

We employ $x < y$ as the usual shorthand: $x \leq y$ and $x \neq y$. We denote as $x \prec y$ the fact that x is an immediate predecessor of y in $\mathcal{L}$, that is, $x < y$ and, for all z, $x < z \leq y$ implies $z = y$ (equivalently, $x \leq z < y$ implies $x = z$).

We focus on algorithms that have access to an underlying finite lattice $\mathcal{L}$ of size $|\mathcal{L}| = n$, with ordering denoted $\leq$; abusing language slightly, we denote by $\mathcal{L}$ as well its carrier set. The *width* $w(\mathcal{L})$ of the lattice $\mathcal{L}$ is the maximum size of an antichain (a subset of $\mathcal{L}$ formed by pairwise incomparable elements). The lattice is assumed to be available for our algorithms in the form of an abstract

data type offering an iterator that traverses all the elements of the carrier set, together with the operations of testing for the ordering (given $x, y \in \mathcal{L}$, find out whether $x \leq y$) and computing the meet $x \wedge y$ and join $x \vee y$ of $x, y \in \mathcal{L}$; also the constants $\top \in \mathcal{L}$ and $\bot \in \mathcal{L}$ are assumed available.

The algorithms we consider are to perform the task of constructing explicitly the Hasse diagram (also known as the reflexive and transitive reduction) of the given lattice: $H(\mathcal{L}) = \{(x, y) \mid x \prec y\}$. By projecting the Hasse diagram along the first or the second component we find our crucial ingredients: the well-known upper and lower covers.

Definition 1. *The upper cover of* $x \in \mathcal{L}$ *is* $\mathrm{uc}(x) = \{y \mid x \prec y\}$. *The lower cover of* $y \in \mathcal{L}$ *is* $\mathrm{lc}(y) = \{x \mid x \prec y\}$.

The following immediate fact is stated separately just for purposes of easy later reference:

Proposition 1. *If* $x < y$ *then there is* $z \in \mathrm{uc}(x)$ *such that* $x \prec z \leq y$; *and there is* $z' \in \mathrm{lc}(y)$ *such that* $x \leq z' \prec y$.

We will use as well yet another easy technicality:

Lemma 2. *If* $x_1 \prec y$ *and* $x_2 \prec y$, *with* $x_1 \neq x_2$ *then* $x_1 \vee x_2 = y$.

Proof. Since $y \geq x_1$ and $y \geq x_2$ we have $y \geq x_1 \vee x_2$. Then, $x_1 \neq x_2$ implies that they are mutually incomparable, since otherwise the smallest is not an immediate predecessor of y; this implies that $y \geq x_1 \vee x_2 > x_1$, whence $y = x_1 \vee x_2$ as $x_1 \prec y$. □

3 The Border Algorithm in Lattices

The algorithms we are considering here have in common the fact that they traverse the lattice and explicitly maintain a subset of the elements seen so far: those that still might be used to identify new Hasse edges. This subset is known as the "border" and, as it evolves during the traversal, actually each element $x \in \mathcal{L}$ "gets its own border" associated as the algorithm reaches it. The border associated to an element may be potentially used to construct new edges touching it (although these edges may not touch the border elements themselves): more precisely, operations on the border for x will result in $\mathrm{uc}(x)$, hence in the Hasse edges of the form (x, z).

In previous references the border is defined in terms of the elements already processed, and its properties are mixed with those of the algorithm that uses it. Instead, we study axiomatically the properties of the notion of "border" on itself, always as a function of the element for which the border will be considered as a source of Hasse edges, in a manner that is independent of the fact that one is traversing the lattice. This allows us to clarify which abstract properties are necessary for border-based algorithms, so that we can generalize them to arbitrary lattices, traversed in flexible ways. Our key definition is, therefore:

Definition 2. *Given $x \in \mathcal{L}$ and $B \subseteq \mathcal{L}$, B is a* border for x *if the following properties hold:*

1. $\forall y \in B\,(y \not\leq x)$;
2. $\forall z\,(x \prec z \Rightarrow \exists y \in B\,(y \leq z))$.

That is, x is never above an element of a border, but each upper cover of x is; this last condition is equivalent to: all elements strictly above x are greater than or equal to some element of the border. Since $x \leq (x \vee y)$ always holds and $x = (x \vee y)$ if and only if $y \leq x$, we get:

Lemma 3. *Let B be a border for x. Then $\forall y \in B\,(x < x \vee y)$.*

All our borders will fulfill an extra "antichain" condition; the only use to be made of this fact is to bound the size of every border by the width of the lattice.

Definition 3. *A border B is* proper *if every two different elements of B are mutually incomparable.*

The key property of borders, that shows how to extract Hasse edges from them, is the following:

Theorem 1. *Let B be a border for x_0. For all x_1 with $x_0 < x_1$, the following are equivalent:*

1. $x_1 \in \mathrm{uc}(x_0)$ *(that is, $x_0 \prec x_1$);*
2. *there is $y \in B$ such that $x_1 = (x_0 \vee y)$ and, for all $z \in B$, if $(x_0 \vee z) \leq (x_0 \vee y)$ then $(x_0 \vee z) = (x_0 \vee y)$.*

Proof. Given $x_0 \prec x_1$, we can apply the second condition in the definition of border for x_0: $\exists y \in B\,(y \leq x_1)$. Using Lemma 3, $x_0 < (x_0 \vee y) \leq x_1$, implying $(x_0 \vee y) = x_1$ since $x_0 \prec x_1$. Additionally, assuming $(x_0 \vee z) \leq (x_0 \vee y)$ for some $z \in B$ leads likewise to $x_0 < (x_0 \vee z) \leq (x_0 \vee y) = x_1$ and the same property applies to obtain $(x_0 \vee z) = (x_0 \vee y) = x_1$.

Conversely, again Lemma 3 gives $x_0 < (x_0 \vee y) = x_1$. By Proposition 1, there is $z_0 \in \mathrm{uc}(x_0)$ with $x_0 \prec z_0 \leq (x_0 \vee y) = x_1$. We apply the second condition of borders to $x_0 \prec z_0$ to obtain $z_1 \in B$ with $z_1 \leq z_0$, whence $(x_0 \vee z_1) \leq z_0 \leq (x_0 \vee y) = x_1$, allowing us to apply the hypothesis of this direction: $(x_0 \vee z_1) \leq (x_0 \vee y)$ with $z_1 \in B$ implies $(x_0 \vee z_1) = (x_0 \vee y)$ and, therefore, $(x_0 \vee z_1) = z_0 = (x_0 \vee y) = x_1$. That is, $x_1 = z_0 \in \mathrm{uc}(x_0)$. □

Therefore, given an arbitrary element x_0 of the lattice, any candidate for being an element of its upper cover has to be obtainable as a join between x_0 and a border element ($x_1 = x_0 \vee y$ for some $y \in B$). Moreover, among these candidates, only those that are minimals represent immediate successors: they come from those y where $(x_0 \vee z) \leq (x_0 \vee y)$ implies $(x_0 \vee z) = (x_0 \vee y)$, for all $z \in B$.

3.1 Advancing Borders

There is a naturally intuitive operation on borders; if we have a border B for x, and we use it to compute the upper cover of x, then we do not need B as such anymore; to update it, seeing that we no longer need to forbid the membership of x, it is natural to consider adding x to the border. If we had a proper border, and we wished to preserve the antichain property, the elements to be removed would be exactly the upper cover just computed, as these are, as we argue below, the only elements comparable to x that could be in a proper border. (All elements other than x are mutually incomparable, as the border was proper to start with.)

Definition 4. *Given $x \in \mathcal{L}$ and a border B for x, the standard step for B and x is $B \cup \{x\} - \mathrm{uc}(x)$.*

Note that this is *not* to say that $\mathrm{uc}(x) \subseteq B$; elements of $\mathrm{uc}(x)$ may or may not appear in B. We will apply the standard step always when B is a border for x, but let us point out that the definition would be also valid without this constraint, as it consists of just some set-theoretic operations.

Proposition 2. *Let B be a proper border for x. Then the standard step for B and x is also an antichain.*

Proof. Elements of the standard step different from x and from all elements of $\mathrm{uc}(x)$ were already in the previous proper border and are, therefore, mutually incomparable. None of them is below x, by the first border property. If $y > x$ for some $y \in B$, then $y \geq z \succ x$ for some $z \in B$, and the antichain property of B tells us that $y = z$ so that it gets removed with $\mathrm{uc}(x)$. □

However, we are left with the problem that we have now a candidate border but we lack the lattice element for which it is intended to be a border. In [14] and [5], the algorithm moves on to an intent set of the same cardinality as x, whenever possible, and to as small as possible a larger intent set if all intents of the same cardinality are exhausted. In [20] it is shown that, for their variant of the Border algorithm, it suffices to follow a (reversed) linear embedding of the lattice. Here we follow this more flexible approach, which is easier now that we have stated the necessary properties of borders with no reference to the order of traversal: there is no need of considering intent sets and their cardinalities.

Both lattices and their Hasse diagrams can be seen as directed acyclic graphs, by orienting the inequalities in either direction; here we choose to visualize edges (x, y) as corresponding to $x \leq y$. A linear embedding corresponds to the well-known operation of topological sort of directed acyclic graphs, which we will employ for lattices in a "reversed" way:

Definition 5. *A reverse topological sort of $\mathcal{L}$ is a total ordering $x_1, \ldots, x_n$ of $\mathcal{L}$ such that $x_i \leq x_j$ always implies $j \leq i$.*

All our development could be performed with a standard topological sort, not reversed, that is, a linear embedding of the lattice's partial order. However,

as it is customary in FCA to guide the visualization through the comparison of extents, the algorithms we build on were developed with a sort of "built-in reversal" that we inherit through reversing the topological sort (see the similar discussion in Section 2.1 of [5]). A reversed topological sort must start with $\top$, hence the initialization is easy:

Proposition 3. *$B = \emptyset$ is a border for $\top \in \mathcal{L}$.*

Proof. Both conditions in the definition of border become vacuously true: the first one as $B = \emptyset$ and the second one as the top element has no upper covers. □

Theorem 2. *Let $x_1, \ldots, x_n$ be a reverse topological sort of $\mathcal{L}$. Starting with $B_1 = \emptyset$, define inductively B_{k+1} as the standard step for B_k and x_k. Then, for each k, B_k is a border for x_k.*

For clarity, we factor off the proof of the following inductive technical fact, where we use the same notation as in the previous statement.

Lemma 4. *$B_k \subseteq \{x_1, \ldots, x_{k-1}\}$ and, for all x_j with $j < k$, there is $y \in B_k$ with $y \leq x_j$.*

Proof. For $k = 1$, the statements are vacuously true. Assume it true for k, and consider $B_{k+1} = B_k \cup \{x_k\} - \mathrm{uc}(x_k)$, the standard step for B_k and x_k. The first statement is clearly true. For the second, x_k is itself in B_{k+1} and, for the rest, inductively, there is $y \in B_k$ with $y \leq x_j$. We consider two cases; if $y \notin \mathrm{uc}(x_k)$, then the same y remains in B_{k+1}; otherwise, $x_k \prec y \leq x_j$, and x_k is the corresponding new y in B_{k+1}. □

Proof (of Theorem 2). Again by induction on k; we see that the basis is Proposition 3. Assuming that B_k is a border for x_k, we consider $B_{k+1} = B_k \cup \{x_k\} - \mathrm{uc}(x_k)$. Applying the lemma, $B_{k+1} \subseteq \{x_1, \ldots, x_k\}$, which ensures immediately that $\forall y \in B_{k+1}\, (y \not\leq x_{k+1})$ by the property of the reverse topological sort, and the first condition of borders follows. For the second, pick any $z \in \mathrm{uc}(x_{k+1})$; by the condition of reverse topological sort, z, being a strictly larger element than x_{k+1}, must appear earlier than it, so that $z = x_j$ with $j < k + 1$. Then, again the lemma tells us immediately that there is $y \in B_{k+1}$ with $y \leq x_j = z$, as we need to complete the proof. □

3.2 The Generalized Border Algorithm

The algorithm we end up validating through our theorems has almost the same high-level description as the rendering in [5]; the most conspicuous differences are: first, that a reverse topological sort is used to initialize the traversal of the lattice; and, second, that the "reversed lattice" model in [5] has the consequence that their set-theoretic intersection in computing candidates becomes a lattice join in our generalization. Another minor difference is that Proposition 3 spares us the separate handling of the first element of the lattice.

RevTopSort($\mathcal{L}$);
$B = \emptyset$;
$H = \emptyset$;
for x in $\mathcal{L}$, according to the sort **do**
  candidates = $\{x \vee y \mid y \in B\}$;
  cover = minimals(candidates);
  for z *in* cover **do** add (x, z) to H;
  $B = B \cup \{x\} -$ cover;
end

Algorithm 1. The Generalized Border Algorithm

Theorem 2 and Proposition 3 tell us that the following invariant is maintained: B is a border for x. Then, the Hasse edges are computed and added to H according to Theorem 1, in two steps: first, we prepare the list of joins $x \vee y$ and, then, we keep only the minimal elements in it. In essence, this process is the same as described (in somewhat different renderings) in [5], [14] or [20]; however, while the definition of border given in [20] (and recalled in [5]) leads, eventually, to the same notion employed in this paper, further development of a general algorithm that works outside the formal concept analysis framework is dropped off from [20] on efficiency considerations. Moreover, the border algorithm described in [14] works exclusively on the set of intents and assumes the elements are sorted sizewise. The validations of the algorithms in these references rely very much, at some points, on the fact that the lattice is a sublattice of a powerset and contains formal concepts, explicitly operating set-theoretically on their intents. Theorem 1 captures the essence of the notion of border and lifts the algorithm to arbitrary lattices.

One additional difference comes from the fact that the cost of computing the meet and join operations plays a role in the complexity analysis, but is not available in the general case. If we assume that meet and join operations take constant time, then the total running time of the algorithm (except for the sort initialization, which takes $\mathcal{O}(|\mathcal{L}| \log |\mathcal{L}|)$) is bounded by $\mathcal{O}(|\mathcal{L}| w(\mathcal{L})^2)$. By comparison with [20], one can see that one factor of the formula given in [20] gets dropped under the constant time assumption for computing meet and join. However, this assumption may be unreasonable in certain applications; the same reference indicates that their FCA target case requires a considerable amount of graph search for the same operations. Nevertheless, in absence of further information about the specific lattice at hand, it is not possible to provide a finer analysis.

We must point out that, in our implementation, we have employed a heapsort-based version that keeps providing us the next element to handle by means of an iterator, instead of completing the sorting step for the initialization.

4 Distributivity and the iPred Algorithm

In [5], an extra sophistication is introduced that, as demonstrated both formally in the complexity analysis of the algorithm and also practically, leads to a faster

algorithm; namely, if some further information is maintained along, once the candidates are available there is a constant-time test to pick those that are in the cover, by employing the duality $y \in \mathrm{uc}(x) \Leftrightarrow x \in \mathrm{lc}(y) \Leftrightarrow x \prec y$. Constant time also suffices to maintain the additional information. This gives the iPred algorithm. However, it seems that the unavoidable price is to work on formal concepts, as the extra information is heavily set-theoretic (namely, a union of set differences of previously found cover sets for the candidate under study).

Again we show that a fully abstract, lattice-theoretic interpretation exists, and we show that the essential property that allows for the algorithm to work is distributivity: be it due to a distributive $\mathcal{L}$, or, as in fact happens in iPred, due to the embedding of the lattice into a distributive lattice, in the same way as concept lattices (possibly nondistributive) can be embedded in the distributive powerset lattice.

We start treating the simplest case, of very limited usefulness in itself but good as stepping stone towards the next theorem. The property where distributivity can be applied later, if available, is as follows:

Proposition 4. *Consider two comparable elements, $x < z$, from $\mathcal{L}$; let $Y \subseteq \mathrm{lc}(z)$ be the set of lower covers of z that show up in the reverse topological sort before x (it could be empty). Then, $x \in \mathrm{lc}(z)$ if and only if $\bigwedge_{y \in Y}(x \vee y) \geq z$.*

Proof. Applying Proposition 1, we know that there is some $y \in \mathrm{lc}(z)$ such that $x \leq y \prec z$. Any such y, if different from x, must appear before x in the reverse topological sort.

Suppose first that no lower covers of z appear before x, that is, $Y = \emptyset$. Then, no such y different from x can exist; we have that both $x = y \prec z$ and $\bigwedge_{y \in Y}(x \vee y) = \top \geq z$ trivially hold.

In case Y is nonempty, assume first $x \prec z$; we can apply Lemma 2: $x \vee y = z$ for every $y \in Y$, hence $\bigwedge_{y \in Y}(x \vee y) = z$. To argue the converse, assume $x \notin \mathrm{lc}(z)$ and let $x \leq y' \prec z$ as before, where we know further that $x \neq y'$: then $y' \in Y$, so that $\bigwedge_{y \in Y}(x \vee y) \leq (x \vee y') = y' < z$. □

This means that the test for minimality of Algorithm 1 can be replaced by checking the indicated inequality; but it is unclear that we really save time, as a number of joins have to be performed (between the current element x and all the elements in the lower cover of the candidate z that appeared before x in the reverse topological sort) and the meet of their results computed. However, clearly, in distributive lattices the test can be rephrased in the following, more convenient form:

Proposition 5. *Assume $\mathcal{L}$ distributive. In the same conditions as in the previous proposition, x is in the lower cover of z if and only if $x \vee (\bigwedge_{y \in Y} y) \geq z$.*

This last version of the test is algorithmically useful: as we keep identifying elements $Y = \{y_1, \ldots, y_m\}$ of $\mathrm{lc}(z)$, we can maintain the value of $y = \bigwedge_{i \in \{1,\ldots,m\}} y_i$; then, we can test a candidate z by computing $x \vee y$ and comparing this value to z. Afterwards, we update y to $y \wedge x$ if $x = y_{m+1}$ is indeed in the cover. This may save the loop that tests for minimality at a small price.

However, unfortunately, if the lattice is not distributive, this faster test may fail: given $Y \subseteq \mathrm{lc}(z)$, the cover elements found so far along the reverse topological sort, it is always true that x is in the lower cover of z if $x \vee (\bigwedge_{y \in Y} y) \geq z$, because $z \leq x \vee (\bigwedge_{y \in Y} y) \leq \bigwedge_{y \in Y} (x \vee y)$ and, then, one of the directions of Proposition 4 applies; but the converse does not hold in general. Again an example is furnished by Figure 1(a), one of the basic, standard examples of a small nondistributive lattice; assume that the traversal follows the natural ordering of the labels, and consider what happens after seeing that 1 and 2 are indeed lower covers of $z = \top$. Upon considering $x = 3$, we have $Y = \{1, 2\}$, so that $x \vee (\bigwedge Y) = x \vee \bot = x < z$, yet x is a lower cover of z and, in fact, $\bigwedge_{y \in Y} (x \vee y) = (3 \vee 1) \wedge (3 \vee 2) = \top$. Hence, the distributivity condition is necessary for the correctness of the faster test.

4.1 The Generalized iPred Algorithm

The aim of this subsection is to show the main contribution of this paper: we can spare the loop that tests candidates for minimality in an indirect way, whenever a distributive lattice is available where we can embed $\mathcal{L}$. However, we must be careful in how the embedding is performed: the right tool is an injective homomorphism of join-semilattices. Recall that, often, this will *not* be a lattice morphism. Such an example is the identity morphism having as domain the carrier set of a concept lattice $\mathcal{L}$ over the set of attributes X, and as range, $\mathcal{P}(X)$ (see Section 5 for more details on this particular case).

Theorem 3. *Let $(\mathcal{L}', \leq, \vee)$ be a distributive join-semilattice and $f : \mathcal{L} \to \mathcal{L}'$ an injective homomorphism. Consider two comparable elements, $x < z$, from $\mathcal{L}$; let $Y \subseteq \mathrm{lc}(z)$ be the set of lower covers of z that show up in the reverse topological sort before x. Then, $x \prec z$ if and only if $f(x) \vee (\bigwedge_{y \in Y} f(y)) \geq f(z)$.*

Proof. If $Y = \emptyset$ we have $x \prec z$ as in Proposition 4; for this case, $\bigwedge_{y \in Y} f(y) = \top$ (of $\mathcal{L}'$) and $f(x) \vee (\bigwedge_{y \in Y} f(y)) = f(x) \vee \top = \top \geq f(z)$.

For the case where $Y \neq \emptyset$, assume first $x \prec z$ and apply Proposition 4: we have that $\bigwedge_{y \in Y} (x \vee y) \geq z$ whence $f(\bigwedge_{y \in Y} (x \vee y)) \geq f(z)$. By Lemma 1, we obtain $f(z) \leq f(\bigwedge_{y \in Y} (x \vee y)) \leq \bigwedge_{y \in Y} f(x \vee y) = \bigwedge_{y \in Y} (f(x) \vee f(y)) = f(x) \vee \bigwedge_{y \in Y} f(y)$, where we have applied that f commutes with join and that $\mathcal{L}'$ is distributive.

For the converse, arguing along the same lines as in Proposition 4, assume $x \notin \mathrm{lc}(z)$ and let $x \leq y' \prec z$ with $x \neq y'$ so that $y' \in Y$: necessarily $\bigwedge_{y \in Y} f(y) \leq f(y')$, so that $f(x) \vee (\bigwedge_{y \in Y} f(y)) \leq f(x) \vee f(y') = f(x \vee y') = f(y') < f(z)$, where the last step makes use of injectiveness. □

The generalized iPred algorithm is based on this theorem, which proves it correct. In it, the homomorphism f is assumed available, and table LC keeps, for each z, the meet of the $f(x)$'s for all the lower covers x of z seen so far.

```
RevTopSort(ℒ);
B = ∅;
H = ∅;
for x in ℒ, according to the sort do
    LC[x] = ⊤;
    candidates = {x ∨ y | y ∈ B};
    for z in candidates do
        if f(x) ∨ LC[z] ≥ f(z) then
            add (x, z) to H;
            LC[z] = LC[z] ∧ f(x);
            B = B − {z};
        end
    end
    B = B ∪ {x};
end
```

Algorithm 2. The Generalized iPred Algorithm

In the Appendix below, we provide some example runs for further clarification. Regarding the time complexity, again we lack information about the cost of meets, joins, and comparisons in both lattices, and also about the cost of computing the homomorphism. Assuming constant time for these operations, the running time of the generalized iPred algorithm is $\mathcal{O}(|\mathcal{L}|w(\mathcal{L}))$ (plus sorting): the main loop (line 4-15) is repeated $|\mathcal{L}|$ times, and then for each of the at most $w(\mathcal{L})$ candidates, the algorithm checks if a certain condition is met (in constant time) and updates the diagram and the border in the positive case.

If meets and joins do not take constant time, there is little to say at this level of generality; however, for the particular case of the original iPred, which only works for lattices of formal concepts, see [5]: in the running time analysis there, one extra factor appears since the meet operation (corresponding to a set union plus a closure operation) is not guaranteed to work in constant time.

5 Conclusions and Future Work

We have provided a formal framework for the task of computing Hasse diagrams of arbitrary lattices through the notion of "border associated with a lattice element". Although the concept of *border* itself is not new, our approach provides a different, more "axiomatic" point of view that facilitates considerably the application of this notion to algorithms that construct Hasse diagrams outside the formal concept analysis world.

While Algorithm 1 is a clear, straightforward generalization of the Border algorithm of [20,5] (although the correctness proof is far less straightforward), we consider that we should explain further in what sense the iPred algorithm comes out as a particular case of Algorithm 2. In fact, the iPred algorithm uses set-theoretic operations and, therefore, is operating with sets that do not belong to the closure space: effectively, it has moved out of the concept lattice into the

(distributive) powerset lattice. Starting from a concept lattice $(\mathcal{L}, \leq, \vee, \wedge)$ on a set X of attributes, we can define:

- $x \leq y \Leftrightarrow x \supseteq y$
- $x \vee y := x \cap y$
- $x \wedge y := \bigvee\{z \in \mathcal{L} \mid z \leq x, z \leq y\} = \bigcap\{z \in \mathcal{L} \mid z \supseteq x, z \supseteq y\}$
- $\top := \emptyset, \bot := X$

Thus, $\mathcal{L}$ is a join-subsemilattice of the (reversed) powerset on X, and we can define $f : \mathcal{L} \to \mathcal{P}(X)$ as the identity function: it is injective, and it is a join-homomorphism since $\mathcal{L}$, being a concept lattice, is closed under set-theoretic intersection. Therefore, Theorem 3 can be translated to: $x \in \mathrm{lc}(z)$ if and only if $x \cap (\bigcup_{y \in Y} y) \subseteq z$, where Y is the set of lower covers of z already found; this is fully equivalent to the condition behind algorithm iPred of [5] (see Proposition 1 on page 169 in [5]). Additionally, iPred works on one specific topological sort, where all intents of the same cardinality appear together; our generalization shows that this is not necessary: any linear embedding suffices.

A further application we have in mind refers to various forms of implication known as multivalued dependency clauses [17,18]; in [2,3,4], these clauses are shown to be related to partition lattices in a similar way as implications are related to concept lattices through the Guigues-Duquenne basis ([8,10]); further, certain database dependencies (the degenerate multivalued dependencies of [17,18]) are related to these clauses in the same way as functional dependencies correspond to implications. Data Mining algorithms that extract multivalued dependencies do exist [19] but we believe that alternative ones can be designed using Hasse diagrams of the corresponding partition lattices or related structures like split set lattices [2]. The task is not immediate, as functional and degenerate multivalued dependencies are of the so-called "equality-generating" sort but full-fledged multivalued dependencies are of the so-called "tuple-generating" sort, and their connection to lattices is more sophisticated (see [2]); but we still hope that further work along this lattice-theoretic approach to Hasse diagrams would allow us to create a novel application to multivalued dependency mining.

Appendix: Examples

We exemplify here some runs of iPred, for the sake of clarity. First we see how it operates on the lattice in Figure 1(a), denoted $\mathcal{L}$ here, using as f the injective homomorphism into the distributive lattice of Figure 1(b) provided by the labels. The run is reported in Table 1, where we can see that we identify the respective upper covers of each of the lattice elements in turn. The linear order is assumed to be $(\top, 1, 2, 3, \bot)$. Only the last loop has more than one candidate, in fact three. The snapshots of the values of B, H, and LC reported in each row (except the initialization) are taken at the end of the corresponding loop, so that each reported value of B is a border for the next row. In the Hasse edges H, thin lines represent edges that are yet to be found, and thick lines represent the edges

Table 1. Example run of the iPred algorithm using the lattices in Figure 1

$\mathcal{L}$	B	H	cand	LC[$\top$]	LC[1]	LC[2]	LC[3]	LC[$\bot$]
init	$\emptyset$							
$\top$	$\{\top\}$		$\emptyset$	$\top$				
1	$\{1\}$		$\{\top\}$	1	$\top$			
2	$\{1,2\}$		$\{\top\}$	4	$\top$	$\top$		
3	$\{1,2,3\}$		$\{\top\}$	$\bot$	$\top$	$\top$	$\top$	
$\bot$	$\emptyset$		$\{1,2,3\}$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$

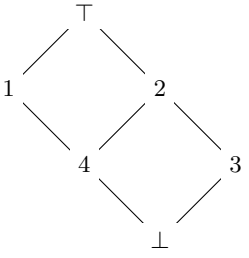


Fig. 2. A distributive lattice

Table 2. Example run of the iPred algorithm on the lattice in Figure 2

$\mathcal{L}$	B	H	cand	LC[$\top$]	LC[1]	LC[2]	LC[3]	LC[4]	LC[$\bot$]
init	$\emptyset$								
$\top$	$\{\top\}$		$\emptyset$	$\top$					
1	$\{1\}$		$\{\top\}$	1	$\top$				
2	$\{1,2\}$		$\{\top\}$	4	$\top$	$\top$			
3	$\{1,3\}$		$\{\top,2\}$	4	$\top$	3	$\top$		
4	$\{3,4\}$		$\{1,2\}$	4	4	$\bot$	$\top$	$\top$	
$\bot$	$\emptyset$		$\{3,4\}$	4	$\bot$	$\bot$	$\bot$	$\bot$	$\top$

found so far. Recall that the values of LC are actually elements of the distributive lattice of Figure 1(b), and not from $\mathcal{L}$.

All along the run we can see that LC[z] indeed maintains the meet of the set of predecessors found so far for $f(z)$ in the distributive embedding lattice; of course, this meet is $\top$ whenever the set is empty.

Let us compare with the run on the distributive lattice in Figure 2, where the homomorphism f is now the identity. Observe that the only different Hasse edge

is the one above 3 which now goes to 2 instead of going to $\top$. Again the linear sort follows the order of the labels.

Due to the similarity among the Hasse diagrams, the run of generalized iPred on this lattice starts exactly like the one already given, up to the point where node 3 is being processed. At that point, 2 is candidate and will indeed create an edge, but 1 leads to candidate $1 \vee 3 = \top$ for which the test fails, as $\mathrm{LC}[\top] = 4$ at that point, and $3 \vee 4 = 2 < \top$. Hence, this candidate has no effect. After this, the visits to 4 and $\bot$ complete the Hasse diagram with their corresponding upper covers.

References

1. Aggarwal, C.C., Yu, P.S.: A new approach to online generation of association rules. IEEE Transactions on Knowledge and Data Engineering 13(4), 527–540 (2001)
2. Baixeries, J.: Lattice Characterization of Armstrong and Symmetric Dependencies. Ph.D. thesis, Universitat Politècnica de Catalunya (2007)
3. Baixeries, J.: A formal context for symmetric dependencies. In: Medina and Obiedkov [15], pp. 90–105
4. Baixeries, J., Balcázar, J.L.: Unified characterization of symmetric dependencies with lattices. In: Ganter, B., Kwuida, L. (eds.) Contributions to the 4th International Conference on Formal Concept Analysis (ICFCA). Verlag Allgemeine Wissensch (2006)
5. Baixeries, J., Szathmary, L., Valtchev, P., Godin, R.: Yet a faster algorithm for building the Hasse diagram of a concept lattice. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS, vol. 5548, pp. 162–177. Springer, Heidelberg (2009)
6. Balcázar, J.L.: Redundancy, deduction schemes, and minimum-size bases for association rules. Logical Methods in Computer Science 6(2:3), 1–33 (2010)
7. Davey, B., Priestley, H.: Introduction to Lattices and Orders, 2nd edn. Cambridge University Press, Cambridge (1991)
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
9. Godin, R., Missaoui, R., Alaoui, H.: Incremental concept formation algorithms based on Galois (concept) lattices. Computational Intelligence 11, 246–267 (1995)
10. Guigues, J., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines 95, 5–18 (1986)
11. Kryszkiewicz, M.: Representative association rules. In: Wu, X., Ramamohanarao, K., Korb, K.B. (eds.) PAKDD 1998. LNCS (LNAI), vol. 1394, pp. 198–209. Springer, Heidelberg (1998)
12. Kuznetsov, S.O., Obiedkov, S.A.: Algorithms for the construction of concept lattices and their diagram graphs. In: Raedt, L.D., Siebes, A. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 289–300. Springer, Heidelberg (2001)
13. Luxenburger, M.: Implications partielles dans un contexte. Mathématiques et Sciences Humaines 29, 35–55 (1991)
14. Martin, B., Eklund, P.W.: From concepts to concept lattice: A border algorithm for making covers explicit. In: Medina and Obiedkov [15], pp. 78–89
15. Medina, R., Obiedkov, S. (eds.): ICFCA 2008. LNCS (LNAI), vol. 4933. Springer, Heidelberg (2008)

16. Nourine, L., Raynaud, O.: A fast algorithm for building lattices. Information Processing Letters 71(5-6), 199–204 (1999)
17. Sagiv, Y., Delobel, C., Parker Jr., D.S., Fagin, R.: An equivalence between relational database dependencies and a fragment of propositional logic. Journal of the ACM 28(3), 435–453 (1981)
18. Sagiv, Y., Delobel, C., Parker Jr., D.S., Fagin, R.: Correction to "An equivalence between relational database dependencies and a fragment of propositional logic". Journal of the ACM 34(4), 1016–1018 (1987)
19. Savnik, I., Flach, P.A.: Discovery of multivalued dependencies from relations. Intelligent Data Analysis 4(3-4), 195–211 (2000)
20. Valtchev, P., Missaoui, R., Lebrun, P.: A fast algorithm for building the Hasse diagram of a Galois lattice. In: Leroux, P. (ed.) Publications du LaCIM, pp. 293–306 (2000)
21. Zaki, M.J., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Transactions on Knowledge and Data Engineering 17(4), 462–478 (2005)

Selecting Important Concepts Using Weights

Radim Belohlavek and Juraj Macko

Department of Computer Science
Palacky University, Olomouc
17. listopadu 12, CZ-77146 Olomouc
Czech Republic
{radim.belohlavek,juraj.macko}@upol.cz

Abstract. We present an approach that enables one to select a reasonable small number of possibly important formal concepts from the set of all formal concepts of a given input data. The problem to select a small number of concepts appears in applications of formal concept analysis when the number of all formal concepts of the input data is large. Namely, a user often asks for a list of "important concepts" in such case. In the present approach, attributes of the input data are assigned weights from which values of formal concepts are determined. Formal concepts with larger values are considered more important. The attribute weights are supposed to be set by the users. The approach is a continuation of our previous approaches that utilize background knowledge, i.e. additional knowledge of a user, to select parts of concept lattices. In addition to the approach, we present illustrative examples.

1 Introduction

1.1 Problem Description

It is well-known from the applications of formal concept analysis (FCA) that even a middle-size input data often contains quite a large set of formal concepts. If the concept lattice, i.e. the hierarchically ordered collection of all formal concepts, is to be directly presented to a user, the large number of formal concepts presents a problem. Note in passing that the problem of a large number of patterns is a general problem faced in many data analysis methods that look for certain types of patterns in data. Note also that in many situations, the concept lattice is not directly presented to a user but, instead, is used for further data preprocessing, in which case the large number of all formal concepts may not be a problem.

1.2 Outline of Our Approach

Quite often, a user naturally considers some formal concepts important (or natural) and considers other formal concepts not important (or not natural). Therefore, it is reasonable to present to the user only those formal concepts considered by him as important. Alternatively, importance of formal concepts may be considered as a graded phenomenon—some concepts are more important than other

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 65–80, 2011.

concepts which themselves are more important than other ones, etc. The basic hypothesis behind our approach is that user's judgment regarding importance of formal concepts is based on his background knowledge. A background knowledge is a knowledge regarding the objects and attributes that is different from the one represented by the input formal context, namely it is supplementary to the knowledge given by the formal context.

This idea was investigated in some previous papers of the first author, see e.g. [2,3,4]. In particular, [2,4] explores the case when the background knowledge concerns importance of attributes. This type of information is then reflected in the user's assessment of importance of formal concepts of the formal context. As an example, consider a formal context in which objects are certain organisms and attributes are some of their features. Suppose that among the attributes are "warm-blooded" and "red" (referring to the color of organisms). The attribute concepts generated by these attributes, i.e. the formal concepts that can verbally be described as *warm-blooded organisms* and *red organisms*, are (in non-trivial cases) two different formal concepts which have the same status, namely, both are elements of the concept lattice presented to a user. However, from an expert user point of view, *warm-blooded organisms* is certainly more important than *red organisms*. Note that a different but the same type of example, which occured in experiments with FCA of fossils, led to the idea of background knowledge investigated in [2,4] and that [1] provides an application of FCA with this type of background knoweldge to fossil data.

In [2,4], a background knowledge concerning importance of attributes is represented by a set of formulas, called attribute-dependency formulas (AD formulas). An example of such a formula is:

$$\text{red} \sqsubseteq \text{warm-blooded} \sqcup \text{cold-blooded}.$$

A formal concept is then considered important if it satisfies the following condition: if red belongs to the intent then either warm-blooded or cold-blooded belongs to the intent of the concept. Therefore, the attribute concept "red organisms" (whose intent presumably does contain neither warm-blooded nor cold-blooded) is not considered important. This way, a background knowledge represented by a set of AD formulas may rule out unimportant formal concepts.

Modeling a background knowledge regarding importance of attributes by AD formulas may be considered a relationally based approach. One may, however, consider also a numerical approach based on assigning weights to attributes. This possibility is explored in the present paper.

Assigning weights to attributes is popular in various methods of data analysis because of its appeal and seeming simplicity: A user assigns weights which are further processed in a way that accordingly treats attributes with higher weights as more important. On the other hand, weights entail a non-trivial problem. Namely, how to assign them? If attribute y_1 is more important than y_2, what values $w(y_1)$ and $w(y_2)$ shall be assigned to y_1 and y_2? Clearly, we want $w(y_1) > w(y_2)$ but what is the appropriate relationship of $w(y_1)$ to $w(y_1)$? Should $w(y_1) = cw(y_2)$ or $w(y_1) = w(y_2) + c$? The problem at the core of these questions is

that in order for the method to be reasonable for a user, the values of weights and the way weights are processed need to have a clear meaning. This is an important moral from the measurement theory [13]. It is exactly these problems why in our previous work, we resorted to relationally based approach and to AD formulas. However, the appeal of weights and their widespread use throughout data analysis motivate us to investigate them within FCA.

In the framework using weights, attribute "warm-blooded" is to be assigned a large weight, because it is an important one, while "red" is to be assigned a small weight, because it is not very important (from a particular user point of view). Doing so, formal concept "red organisms" is to be considered not important because it is generated by an attribute with a low weight (low importance). This is the essence of the approach presented in this paper.

1.3 Related Work

Formal concept analysis offers various methods to deal with large concept lattices. The best known are perhaps the nested line diagrams [10] which represent a kind of a folding/unfolding to present a large concept lattice to a user. A different possibility to handle large concept lattices is to use the various decomposition constructions described in [10]. However, computational properties of these decompositions have, by and large, not yet been investigated. The idea of using background knowledge to reduce the number of patterns presented to a user appears in several forms in data mining [5]. In FCA, a type of background knowledge has been studied in [9] for the for the purpose of attribute exploration. Both the type of the background knowledge and the aims in [9] are different from those which are discussed in our paper. Related to the presented approach are the approaches presented in [2,3,4,7,8,12,14], some of which are discussed elsewhere in this paper. The difference of the presented approach from these approaches is the use of attribute weights which is explored in the present approach.

1.4 Preliminaries and Notation

We assume that the reader is familiar with basic notions of formal concept analysis [10]. A formal context is denoted by $\langle X, Y, I\rangle$. Formal concepts of $\langle X, Y, I\rangle$ are denoted by $\langle A, B\rangle$. A pair $\langle A, B\rangle$ consisting of $A \subseteq X$ and $B \subseteq Y$ is called a formal concept if and only if $A^{\uparrow} = B$ and $B^{\downarrow} = A$ where

$$A^{\uparrow} = \{y \in Y \mid \text{ for each } x \in X \,:\, \langle x, y\rangle \in I\},$$
$$B^{\downarrow} = \{x \in X \mid \text{ for each } y \in Y \,:\, \langle x, y\rangle \in I\}$$

are the set of all attributes common to all objects from A and the set of all objects having all the attributes from B, respectively. The set of all formal concepts of $\langle X, Y, I\rangle$ is denoted by $\mathcal{B}(X, Y, I)$. $\mathcal{B}(X, Y, I)$ equipped with a subconcept-superconcept partial order $\leq$ is the concept lattice of $\langle X, Y, I\rangle$.

2 Weights to Select Important Formal Concepts

In the approach described in this section, attributes are assigned weights from a partially ordered set W of weights. Every formal concept is then assigned a value from a partially ordered set V of values. The values are computed from the weights of the attributes present in the intent of the formal concept. This induces an ordering of formal concepts by values. A threshold may in the end determine a set of "important" formal concepts, i.e. those having at least the threshold value.

2.1 Weights of Attributes

We assume that the *weights* to be assigned to attributes form a partially ordered set $\langle W, \leq \rangle$ (weights are elements $w \in W$). A popular case used in various methods of data analysis as well as in our paper is $\langle W, \leq \rangle = \langle \mathbf{R}, \leq \rangle$ (weights are reals). Given a formal context $\langle X, Y, I \rangle$ and a partially ordered set $\langle W, \leq \rangle$ of weights, we assume that a user assigns weights from W to attributes from Y, i.e. that a user specifies a function

$$w : Y \to W.$$

The weights are supposed to be set according to expert knowledge. The rule of thumb is that the more important the attribute, the larger the weight assigned to it. However, as is mentioned in Section 1.2, without any further considerations, setting the weights may turn to be an ad hoc process of which the user may easily lose control. Some considerations along this line are presented in Section 4.1.

As an example, consider the formal context in Table 1. Table 2 shows an assignment of weights to the attributes reflecting an expert opinion that genus is the most important, habitat is second, and the other attributes are equally important and are of low importance compared to genus and habitat.

Table 1. Formal context of felines

	genus				habitat				size			fur		color			
	Acinonyx	Felis	Leptailurus	Panthera	Africa	America	Asia	Europe	small	medium	large	stripes	spots	black	sandy	white	yellow
Cheetah	×				×					×		×	×				×
Cougar				×		×					×				×		
Jaguar				×		×					×		×	×			×
Lion				×	×						×				×		
Panther				×	×		×			×			×				×
Serval			×		×				×			×	×		×		×
Tiger				×			×				×	×				×	×
Wildcat		×			×	×	×	×	×			×	×	×	×		×

Table 2. Assignment of weights to attributes

	genus				habitat				size			fur		color			
	Acinonyx	Felis	Leptailurus	Panthera	Africa	America	Asia	Europe	small	medium	large	stripes	spots	black	sandy	white	yellow
weight	100	100	100	100	10	10	10	10	1	1	1	1	1	1	1	1	1

2.2 Values of Formal Concepts

As is mentioned above, we want to assign values from a partially ordered set $\langle V, \leq \rangle$ to formal concepts. (In examples we use $W = V = \mathbf{R}$ in this paper; in general, however, weights are of different type than values.) A value of a formal concept $\langle A, B \rangle$ is to be determined by the weights $w(y)$ of attributes $y \in B$ by a suitable *aggregation function*

$$\mathsf{A} : \bigcup_{k=1}^{\infty} W^k \to V.$$

We assume that the restrictions $\mathsf{A}^{(k)} : W^k \to V$ of A to W^k satisfy the following conditions:

$$\mathsf{A}^{(k)} \text{ is isotone for each } k, \tag{1}$$

$$\mathsf{A}^{(k)}(0_W, \ldots, 0_W) = 0_V \text{ and } \mathsf{A}^{(k)}(1_W, \ldots, 1_W) = 1_V \text{ for each } k, \tag{2}$$

$$\mathsf{A}^{(1)}(w) = \mathsf{A}^{(2)}(w, w) = \cdots = \mathsf{A}^{(k)}(w, \ldots, w) = \cdots \tag{3}$$

The conditions in (2) are assumed whenever 0_W, 1_W, 0_V, and 1_V exist (the least and greatest elements of W and V, respectively). These requirements are in accordance with [11].

An aggregation function A enables us to assign a *value* to a formal concept $\langle A, B \rangle$. Informally, $\langle A, B \rangle$ is considered important if it is determined by a set D attributes for which the A-aggregation of their weights results in a high value. However, which set D of attributes is to be taken for $\langle A, B \rangle$ is not immediate. One can take $D = B$, i.e. the intent of $\langle A, B \rangle$. Alternatively, one can take D to be a generator of $\langle A, B \rangle$ with a high value. Yet another option is to consider minimal generators of $\langle A, B \rangle$ as candidates for D. Before considering these options, let for a set $D = \{y_{i_1}, \ldots, y_{i_k}\} \subseteq Y$ of attributes put

$$v(D) = \mathsf{A}(w(y_{i_1}), \ldots, w(y_{i_k})).$$

$v(D)$ is called the value of D. In most cases, $\mathsf{A}(w(y_{i_1}), \ldots, w(y_{i_k}))$ does not depend on the order of the arguments and $v(D)$ is thus defined correctly. In the possible other cases, one needs to define $v(D)$ in an appropriate way. $v(\emptyset)$ may be defined to be 0_V or may be handled ad hoc because of the specificity of $\emptyset$.

As an example, the following functions are aggregation functions:

$$\begin{aligned}\mathsf{A}(w_1,\ldots,w_k) &= \min(w_1,\ldots,w_k),\\ \mathsf{A}(w_1,\ldots,w_k) &= \frac{w_1+\cdots+w_k}{k},\\ \mathsf{A}(w_1,\ldots,w_k) &= \max(w_1,\ldots,w_k).\end{aligned}$$

Let us first consider the option $D = B$, i.e. computing the value of $\langle A, B\rangle$ using the weights of all attributes of the intent B by

$$v_{\mathrm{Int}}(A, B) = v(B).$$

This approach suffers the following drawback. Suppose $B = \{y_1, y_2\}$ where y_1 and y_2 are attributes with a high and low weight (such as y_1 being "warm-blooded" and y_2 being "red") and suppose that $B = \{y_1\}^{\downarrow\uparrow}$. This situation may be interpreted as follows. The "core" of the formal concept consists of y_1 (so the formal concept would naturally be termed "warm-blooded organisms") but since (in our particular formal context) every warm-blooded organism is red, the intent B contains "red" as an accompanying attribute of low importance. In determining the value of formal concepts, e.g. by A being the arithmetic mean, the accompanying attributes lower the value of a formal concept and should be disregarded (in our case, $v_{\mathrm{Int}}(A, B) = \frac{w(y_1)+w(y_2)}{2}$ while an intuitively reasonable way is to take $v(A, B) = w(y_1)$).

Therefore, taking generators instead appears to be a reasonable choice. Recall that for a formal concept $\langle A, B\rangle \in \mathcal{B}(X, Y, I)$, the sets of *generators* and *minimal generators* of $\langle A, B\rangle$ are defined by

$$\begin{aligned}\mathrm{gen}(A, B) &= \{D \subseteq Y \mid D^{\downarrow\uparrow} = B\},\\ \mathrm{mgen}(A, B) &= \{D \in \mathrm{gen}(A, B) \mid E^{\downarrow\uparrow} \subset B \text{ for every } E \subset D\},\end{aligned}$$

respectively. Generators and minimal generators make it possible to define the following two kinds of value:

$$\begin{aligned}v_{\mathrm{gen}}(A, B) &= \max\{v(D) \mid D \in \mathrm{gen}(A, B)\},\\ v_{\mathrm{mgen}}(A, B) &= \max\{v(D) \mid D \in \mathrm{mgen}(A, B)\}.\end{aligned}$$

Clearly,

$$v_{\mathrm{gen}}(A, B) \geq v_{\mathrm{mgen}}(A, B) \text{ and } v_{\mathrm{gen}}(A, B) \geq v_{\mathrm{Int}}(A, B).$$

An argument similar to the one we used to show a possible deficiency of v_{Int} might suggest that v_{gen} is more appropriate than v_{mgen} (this is seen on the concept with intent consisting of G_Pan and S_Lar in Table 3, which is an important one given the weights, but v_{mgen} gives a very small value 1). However, we keep both v_{gen} and v_{mgen} as reasonable choices.

2.3 Selecting Formal Concepts

A choice of an aggregation function A (min, arithmetic mean, max, etc.) and a function v (v_{gen}, v_{mgen}, or v_{Int} if the above drawback does not play a role)

determine an ordering of formal concepts from $\mathcal{B}(X, Y, I)$ according to the values from V (a total order if $\langle V, \leq \rangle$ is totally ordered).

A given value $\theta \in V$ (*threshold value*, selected by a user beforehand or after he sees the values of formal concepts) determines the part

$$\mathcal{B}_\theta(X, Y, I) = \{\langle A, B \rangle \in \mathcal{B}(X, Y, I) \,|\, v(A, B) \geq \theta\}$$

of the whole concept lattice consisting of formal concepts whose value is at least θ. $\mathcal{B}_\theta(X, Y, I)$ may be seen as the set of important formal concepts.

Depending on the aggregation function A and the value-assignment function v, $\mathcal{B}_\theta(X, Y, I)$ may form a substructure of $\mathcal{B}(X, Y, I)$. For example, for $v = v_{\text{Int}}$, if $\mathsf{A} = \min$, $\mathcal{B}_\theta(X, Y, I)$ with the largest formal concept forms a $\bigvee$-sublattice of $\mathcal{B}(X, Y, I)$; if $\mathsf{A} = \max$, $\mathcal{B}_\theta(X, Y, I)$ forms a $\bigwedge$-sublattice of $\mathcal{B}(X, Y, I)$. An investigation of this type of questions is beyond the scope of this paper.

2.4 Comparing Weights to AD Formulas

Consider now the connection of the presented approach to the approach based on AD formulas [2,4]. We show that these two approaches have a different power w.r.t. selecting important concepts. In a sense, this is not surprising because the rationale behind the two approaches are different. Concrete examples are presented in Section 3.

Recall that an *AD formula* over a set Y of attributes is an expression $A \sqsubseteq B$, where $A, B \subseteq Y$. $A \sqsubseteq B$ is true in $M \subseteq Y$ if whenever $A \cap M \neq \emptyset$, then $B \cap M \neq \emptyset$; $A \sqsubseteq B$ is true in a formal concept $\langle C, D \rangle$ if it is true in D (see Section 1.2 for an example). Given a set T of AD formulas over Y and a formal context $\langle X, Y, I \rangle$, the concept lattice constrained by T is denoted by $\mathcal{B}_T(X, Y, I)$ and consists of formal concepts of $\langle X, Y, I \rangle$ in which all AD formulas from T are true, i.e.

$$\mathcal{B}_T(X, Y, I) = \{\langle C, D \rangle \in \mathcal{B}(X, Y, I) \,|\, \text{ each } A \sqsubseteq B \in T \text{ is true in } \langle C, D \rangle\}.$$

The following proposition shows that constraints by weights are not representable by constraints by AD formulas.

Proposition 1. *For $\langle W, \leq \rangle = \langle V, \leq \rangle = \langle \mathbf{R}, \leq \rangle$, A being the arithmetic mean, and v being any of v_{Int}, v_{gen}, or v_{mgen}, there exists a formal context $\langle X, Y, I \rangle$ and a threshold θ such that no set T of AD formulas satisfies $\mathcal{B}_\theta(X, Y, I) = \mathcal{B}_T(X, Y, I)$.*

Proof. Sketch: Consider the following formal context:

I	y_1	y_2	y_3
x_1	×	×	×
x_2	×	×	
x_3	×		×
x_4		×	×

Let $w(y_1) = 10$, $w(y_2) = 1$, $w(y_3) = 1$, and $\theta = 5$. One may easily verify that among the formal concepts in $\mathcal{B}(X, Y, I)$ are $\{y_1, y_2\}$, $\{y_1, y_3\}$, and $\{y_1, y_2, y_3\}$. Moreover, for v being any of v_{Int}, v_{gen}, or v_{mgen}, we have $v(\{y_1, y_2\}) = 5.5$, $v(\{y_1, y_3\}) = 5.5$, and $v(\{y_1, y_2, y_3\}) = 4$. Thus, $\mathcal{B}_\theta(X, Y, I)$ contains $\langle\{y_1, y_2\}^\downarrow, \{y_1, y_2\}\rangle$ and $\langle\{y_1, y_3\}^\downarrow, \{y_1, y_3\}\rangle$, but does not contain $\langle\{y_1, y_2, y_3\}^\downarrow, \{y_1, y_2, y_3\}\rangle$. Now for any T, the system of sets $M \subseteq Y$ in which all formulas from T is an interior system [4]. Therefore, it follows that if $\langle C_1, D_1\rangle, \langle C_2, D_2\rangle \in \mathcal{B}_T(X, Y, I)$ and if $D_1 \cup D_2$ is an intent, then $\langle(D_1 \cup D_2)^\downarrow, D_1 \cup D_2\rangle \in \mathcal{B}_T(X, Y, I)$. Hence, if $\mathcal{B}_\theta(X, Y, I) = \mathcal{B}_T(X, Y, I)$ then since $\langle\{y_1, y_2\}^\downarrow, \{y_1, y_2\}\rangle, \langle\{y_1, y_3\}^\downarrow, \{y_1, y_3\}\rangle \in \mathcal{B}_T(X, Y, I)$, we must have $\langle\{x_1\}, \{y_1, y_2, y_3\}\rangle \in \mathcal{B}_T(X, Y, I)$, a contradiction to $\langle\{x_1\}, \{y_1, y_2, y_3\}\rangle \notin \mathcal{B}_\theta(X, Y, I)$. □

The next proposition shows that constraints by AD formulas are not representable by constraints by weights.

Proposition 2. *There exists a set T of AD formulas and a formal context $\langle X, Y, I\rangle$ such that no totally ordered $\langle W, \leq\rangle$, totally ordered $\langle V, \leq\rangle$, A, v being any of v_{Int}, v_{gen}, or v_{mgen}, and θ satisfy $\mathcal{B}_T(X, Y, I) = \mathcal{B}_\theta(X, Y, I)$.*

Proof. Sketch: Let $T = \{\{y_1\} \sqsubseteq \{y_2\}, \{y_2\} \sqsubseteq \{y_1\}\}$ and consider the "non-equality" formal context $\langle Y, Y, \neq\rangle$ (i.e., $\langle x, y\rangle \in I$ means $x \neq y$ for $x, y \in Y$). Then, every subset of Y is an intent. Both formulas from T are true in $\{y_1, y_2\}$, $\{y_1\} \sqsubseteq \{y_2\}$ is not true in $\{y_1\}$, and $\{y_2\} \sqsubseteq \{y_1\}$ is not true in $\{y_2\}$. Therefore, $\mathcal{B}_T(X, Y, I)$ contains the corresponding formal concept $\langle\ldots, \{y_1, y_2\}\rangle$, but does contain neither $\langle\ldots, \{y_1\}\rangle$ nor $\langle\ldots, \{y_2\}\rangle$. Suppose $\mathcal{B}_T(X, Y, I) = \mathcal{B}_\theta(X, Y, I)$. Because $\{y_1\}$, $\{y_2\}$, and $\{y_1, y_2\}$ are the only generators of the corresponding formal concepts, we have $v(\langle\ldots, \{y_1, y_2\}\rangle) = v(\{y_1, y_2\}) = \mathsf{A}(w(y_1), w(y_2))$, $v(\langle\ldots, \{y_1\}\rangle) = \mathsf{A}(w(y_1))$, and $v(\langle\ldots, \{y_2\}\rangle) = \mathsf{A}(w(y_2))$. Now, $\mathcal{B}_T(X, Y, I) = \mathcal{B}_\theta(X, Y, I)$ implies that $\mathsf{A}(w(y_1), w(y_2)) \geq \theta$, $\mathsf{A}(w(y_1)) < \theta$, and $\mathsf{A}(w(y_2)) < \theta$. This yields a contradiction because, as is easily seen, $\mathsf{A}(w(y_1), w(y_2))$ must lie between $\mathsf{A}(w(y_1))$ and $\mathsf{A}(w(y_2))$. □

3 Illustrative Examples and Experiments

First experiment. Consider the formal context in Table 1. The corresponding concept lattice contains 35 formal concepts and is depicted in Fig. 1. The concept lattice uses the usual labeling [10]. To save space, we use abbreviations of attribute names for labeling, e.g. G_Pan for "genus Panthera", H_Af for "habitat Africa", and the like. As one can see, the concept lattice contains intuitively important formal concepts such as the attribute concept of G_Pan (felines of genus Panthera). However, it also contains formal concepts such as the attribute concept of C_yell (felines with yellow color) which may be regarded as not important because a reasonable classification of felines is not likely to be based on color (color is intuitively an attribute of secondary importance). On the other hand, attribute C_yell may be part of a subconcept of an important concept, such as the formal concept whose intent consists of G_Pan and C_yell (felines of

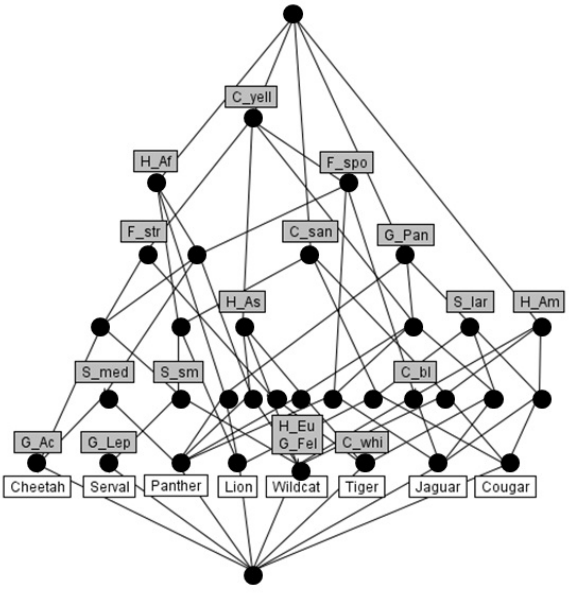


Fig. 1. Concept lattice of formal context from Table 1

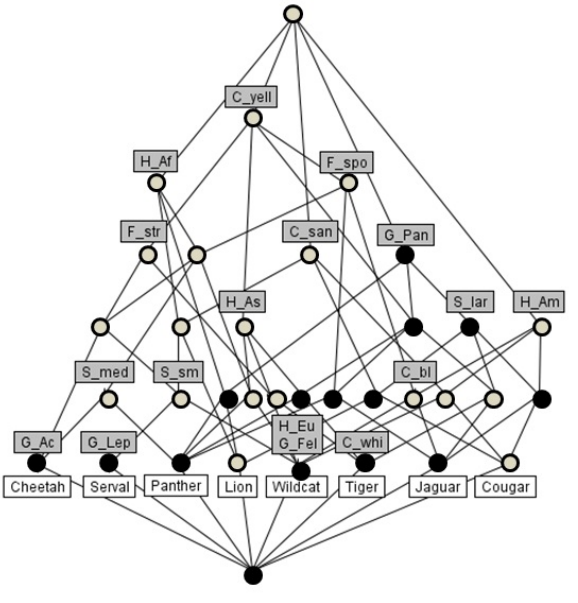


Fig. 2. Formal concepts of the concept lattice from Fig. 1 with $v_{\text{gen}} \geq 50$ (black nodes)

Table 3. Values of formal concepts of the concept lattice from Fig. 1 corresponding to the weights from Table 2

attributes of intent	v_{Int}	v_{gen}	v_{mgen}
G_Pan	100.00	100.00	100.00
G_Fel H_Af H_Am H_As H_Eu S_sm F_str F_spo C_bl C_san C_yell	13.27	100.00	100.00
all attributes	26.41	100.00	100.00
G_Lep H_Af S_sm F_str F_spo C_san C_yell	16.43	100.00	100.00
G_Ac H_Af S_med F_str F_spo C_yell	19.00	100.00	100.00
G_Pan H_As C_yell	37.00	55.00	55.00
G_Pan H_Af	55.00	55.00	55.00
G_Pan H_Am S_lar	37.00	55.00	55.00
G_Pan F_spo C_yell	34.00	50.50	50.50
G_Pan S_lar	50.50	50.50	1.00
G_Pan C_yell	50.50	50.50	50.50
G_Pan S_lar C_san	34.00	50.50	50.50
G_Pan H_As S_lar F_str C_whi C_yell	19.00	50.50	50.50
G_Pan H_Am S_lar F_spo C_bl C_yell	19.00	50.50	50.50
G_Pan H_Af H_As S_med F_spo C_yell	20.50	50.50	50.50
G_Pan H_Am S_lar C_san	28.00	37.00	37.00
G_Pan H_Af S_lar C_san	28.00	37.00	37.00
G_Pan S_lar C_yell	34.00	34.00	1.00
H_As C_yell	5.50	10.00	10.00
H_Af H_As F_spo C_yell	5.50	10.00	10.00
H_Af	10.00	10.00	10.00
H_Am	10.00	10.00	10.00
H_Af F_str F_spo C_yell	3.25	5.50	5.50
H_Af C_san	5.50	5.50	5.50
H_Am C_san	5.50	5.50	5.50
H_Af S_sm F_str F_spo C_san C_yell	2.50	5.50	1.00
H_Af F_spo C_yell	4.00	5.50	5.50
H_As F_str C_yell	4.00	5.50	5.50
H_Am F_spo C_bl C_yell	3.25	5.50	5.50
H_Af S_med F_spo C_yell	3.25	5.50	1.00
F_spo C_yell	1.00	1.00	1.00
C_san	1.00	1.00	1.00
F_str C_yell	1.00	1.00	1.00
C_yell	1.00	1.00	1.00

genus Panthera that are yellow). This subconcept may be regarded as important because it helps us structure an important concept (felines of genus Panthera).

Setting the attribute weights according to Table 2, we obtain the values of formal concepts, v_{Int}, v_{gen}, and v_{mgen}, as depicted in Table 3. The formal concepts are ranked according to v_{gen}. Fig. 2 shows all formal concepts with v_{gen} greater than or equal to $\theta = 50$ (those are represented by black nodes), i.e. shows $\mathcal{B}_\theta(X, Y, I)$. The value corresponding to v_{gen} provides a reasonable assessment of importance of formal concepts. Both v_{Int} and v_{mgen} may be seen as

Table 4. Second assignment of weights to attributes

	genus				habitat				size			fur		color			
	Acinonyx	Felis	Leptailurus	Panthera	Africa	America	Asia	Europe	small	medium	large	stripes	spots	black	sandy	white	yellow
weight	10	10	10	10	100	100	100	100	1	1	1	1	1	1	1	1	1

Table 5. Values of formal concepts of the concept lattice from Fig. 1 corresponding to the weights from Table 4

attributes of intent	v_{Int}	v_{gen}	v_{mgen}
G_Fel H_Af H_Am H_As H_Eu S_sm F_str F_spo C_bl C_san C_yell	37.82	100.00	100.00
H_As C_yell	50.50	100.00	100.00
H_Af H_As F_spo C_yell	50.50	100.00	100.00
H_Af	100.00	100.00	100.00
H_Am	100.00	100.00	100.00
all attributes	26.41	82.00	70.00
G_Pan H_Af H_As S_med F_spo C_yell	35.50	70.00	70.00
G_Pan H_As C_yell	37.00	55.00	55.00
G_Pan H_Af	55.00	55.00	55.00
G_Lep H_Af S_sm F_str F_spo C_san C_yell	16.43	55.00	10.00
G_Ac H_Af S_med F_str F_spo C_yell	19.00	55.00	10.00
G_Pan H_Am S_lar	37.00	55.00	55.00

not appropriate in this example. Namely, as is mentioned above, v_{Int} takes into account weights of all attributes of an intent rather than those weights only that belong to an important "core" of the concept. Hence, the second formal concept in Table 3 is assigned value 13.27 by v_{Int} while it is assigned value 100 by both v_{gen} and v_{mgen} because the "core" that determines the value is {G_Fel} in both cases. The tenth formal concept shows that v_{mgen} may not be appropriate either, because a minimal generator may consist of attributes with low weights (such as S_lar in this case) while there may exist a non-minimal generator with large weights. Tables 4 and 5 show a different setting of weights and the corresponding concepts ranked by v_{gen}. Due to lack of space, we include only formal concepts with $v_{\text{gen}} \geq 55$.

Second experiment. Table 6 contains a formal context for our second experiment. The data consists of 26 belemnite species (objects) and 24 rostrum characteristics (attributes). Due to lack of space, we omit the other 14 characteristics that are used in [1] and provide a smaller data for illustration. Note that in [1], important formal concepts were selected from the 26×38 data by means of AD formulas in an experiment involving an expert paleontologist.

Table 6. Formal context of belemnites

	Size														Flatt			Alveola				Cross				
	large guards (more 65mm)	medium guards (65-80mm)	small guards (less 65mm)	cigar shape in dorsovenral view	lanceolate in DV view	slightly lanceolate in DV view	subcylindrical in DV view	conical in DV view	cigar shape in lateral view	lanceolate in L view	slightly lanceolate in L view	subcylindrical in L view	conical in L view		laterally flattened	dorsally flattened	ventrally flattened	high conical alveolar fracture	low conical alveolar fracture	shallow pseudoalveolus (less 3mm)	deep pseudoalveolus (more 3mm)	oval cross section of alveolar fracture	oval to triangular cross section of AF	triangular cross section of AF	circular cross section of AF
	a	b	c	d	e	f	g	h	i	j	k	l	m		n	o	p	q	r	s	t	u	v	w	x
Actinocamax			×	×					×						×			×							×
P. primus	×	×	×	×	×	×	×			×	×	×					×		×			×			
P. plenus	×	×	×		×	×	×				×	×					×		×			×	×		
P. plenus cf. st.		×			×					×							×		×			×	×		
P. triangulus	×	×	×		×						×	×					×				×			×	
P. aff triangulus	×					×						×					×				×		×		
P. sozhensis	×	×	×		×						×	×					×	×				×			
P. contractus		×	×		×						×	×					×		×			×	×		
P. planus		×			×						×	×					×		×			×			
P. coronatus		×					×						×				×				×	×			
P. matesovae		×			×							×					×			×		×			
P. medwedicicus		×		×			×					×					×				×		×		
P. sp. 1		×	×			×	×					×					×				×		×		
P. sp. 2		×			×							×					×				×	×	×		
P. strehlensis		×	×		×					×	×						×		×			×	×		
P. bohemicus		×				×	×					×	×				×		×			×	×	×	
P. aff. bohemicus		×				×	×					×					×		×			×	×		
P. cobbani		×			×	×					×	×					×		×			×	×		
P. manitobensis	×	×				×					×	×					×		×	×			×	×	
P. cf. manitob.	×					×											×		×	×		×		×	
P. sternbergi	×	×				×					×						×		×			×	×		
P. walkeri		×				×	×				×	×					×		×			×	×		
G. intermedius		×	×		×	×					×	×					×				×		×		
G. surensis			×	×	×	×						×					×			×	×		×		
G. christenseni		×				×						×				×	×				×	×			
G. lundgreni		×				×						×				×	×				×	×			

Table 7. Assignment of weights to attributes

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
weight	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	100	100	100	100	1	1	1	1

Table 7 shows an assignment of weights to the attributes that is based on an assessment of attribute importance provided by the expert [1]. Table 8 contains the corresponding formal concepts with $v_{\text{gen}} > 50$. The last column indicates by $*$ that a formal concept with the same set of attributes from a–x was listed among the important concepts explicitly described by the expert paleontologist in [1]. Since we include only 24 most important attributes in our experiment, the experiment and comparison to [1] needs to be redone with all the 36 belemnite attributes and needs to involve the expert paleontologist. Such experiment is left for an extended version of this paper.

4 Further Issues and Conclusions

4.1 How To Set Weights?

As is mentioned above, a practical problem in setting the weights is that while a user usually has an idea about whether a particular attribute y_1 is more important than another one, say y_2, it might be difficult for the user to decide which weights to assign to the attributes and to justify the decision. The question is in fact the question of meaning of the weights. In some situations, the weights may be given, e.g. as "prices" of the attributes. In general, however, answers to this question need to be connected to the way weights are utilized. Taking into account the utilization of weights described above, one may reason as follows.

It is often the case that a user naturally partitions the attributes into groups in such a way that attributes in every particular group are considered equally important. Let the set Y of attributes be partitioned into disjoint sets $Y_1, \ldots, Y_k$, i.e. $\{Y_1, \ldots, Y_k\}$ be a partition of $Y = \{y_1, \ldots, y_n\}$. Let $Y_i = \{y_{i1}, \ldots, y_{in_i}\}$. That is, $n = n_1 + \cdots + n_k$. The experiments in Section 3 provide examples.

Suppose that attributes from Y_i are considered more important than those from Y_{i+1} for $i = 1, \ldots, k-1$. An appealing way to consider importance of formal concepts is the following (consider $k = 3$). The most important formal concepts are those with a determining set D (i.e. a generator, minimal generator, or intent) of attributes that contains no other attributes than those from Y_1 (at least one but possibly more). The second most important are those with D that contains at least one attribute from Y_1, at least one from Y_2, but no other attribute. The "signatures" of these types of concepts are 100 and 110. The importance of concepts may then be assessed according to the following ordering of signatures:

$$100 > 110 > 111 > 101 > 010 > 011 > 001 \text{ and one my add } > 000. \qquad (4)$$

Now, such ordering (or a different one, representing a different user preferences regarding importance of formal concepts) induces inequalities involving weights.

Table 8. Values of formal concepts of belemnites with $v > 50$ to the weights from Table 7

attributes of intent	v_{Int}	v_{gen}	v_{mgen}	[1]
c q	55.00	100.00	100.00	
a f p r s	46.00	100.00	100.00	
c d e f l p s t v	29.00	100.00	100.00	*
p t	55.00	100.00	100.00	
p s	55.00	100.00	100.00	
p r	55.00	100.00	100.00	
a b f k l p r s v w	26.20	70.00	70.00	*
a f p r s u	38.50	67.00	67.00	
c d i n q x	23.50	55.00	55.00	*
b k p r	32.50	55.00	55.00	
b f l o p t u	21.57	55.00	37.00	*
b c e k p r u	21.57	55.00	55.00	
e l p t	32.50	55.00	55.00	
b c e k l p t	22.86	55.00	55.00	
b g p t	32.50	55.00	55.00	
a b c e k l p q u	19.00	55.00	55.00	*
b g m p t u	23.50	55.00	55.00	*
a b c d e f g j k l p r u	16.23	55.00	55.00	*
b e j p r u	23.50	55.00	10.00	*
c l p t	32.50	55.00	55.00	
b l p s	32.50	55.00	55.00	
l p s	40.00	55.00	55.00	
b l p r	32.50	55.00	55.00	
f p s	40.00	55.00	55.00	
b e p r u	26.20	55.00	55.00	
f p r	40.00	55.00	55.00	
f l p t	32.50	55.00	55.00	
e l p s	32.50	55.00	55.00	
b p t	40.00	55.00	55.00	*
d l p t v	26.20	55.00	55.00	
b p r	40.00	55.00	55.00	
b f g l p r	25.00	55.00	55.00	
a l p t	32.50	55.00	55.00	
b f g l m p r v w	18.00	55.00	55.00	*
l p t	40.00	55.00	55.00	
a f p r	32.50	55.00	55.00	
f l p s v	26.20	50.50	50.50	
b p t u	30.25	50.50	50.50	
b p r v	30.25	50.50	50.50	*
p s u	37.00	50.50	50.50	
p r u	37.00	50.50	50.50	
l p t v	30.25	50.50	50.50	
a b c e k l p t w	19.00	50.50	50.50	*
b f l p r v w	20.29	50.50	50.50	

Consider $110 > 101$ and two formal concepts with determining sets D_1 and D_2 with signatures 110 and 101. Let w_i be the weight assigned to all attributes from Y_i. The worst case for which still $v(D_1) > v(D_2)$ is when D_1 contains 1 atrribute from Y_1 and all n_2 attributes from Y_2, while D_2 contains n_1 attributes from Y_1 and 1 attribute from Y_3. This leads to inequality

$$v(D_1) = \frac{w_1 + n_2 w_2}{n_2 + 1} > \frac{n_1 w_1 + w_3}{n_1 + 1} = v(D_2).$$

When setting the weights, such inequalities need to be taken into account. A further investigation of this line of thought is left for future research.

4.2 Future Research

The following list contains problems for future research:

- A further investigation of relational vs. numeric approaches to represent importance of attributes. Section 2.4 and Section 4.1 may be thought as first steps in this direction.
- Efficient computation. We did not consider the problem of computing efficiently the values of formal concepts. Even though some general approaches may be envisioned, such computations are likely to depend on the particular aggregation and value-assignment functions. Particularly important computational problems are the following: Given a value θ, compute the set $\mathcal{B}_\theta(X, Y, I)$ of all formal concepts with value at least θ. Given a number k, compute the top k formal concepts in the list of all formal concepts ordered by their values $v(A, B)$.
- Expressive power of selecting formal concepts using weights. Which types of subsets $\mathcal{B}_\theta(X, Y, I)$ may be selected from $\mathcal{B}(X, Y, I)$?

Acknowledgment. Supported by Grant No. 103/10/1056 of the Czech Science Foundation and by research plan MSM 6198959214.

References

1. Belohlavek, R., Košt'ák, M., Osicka, P.: Reconstruction of belemnite evolution using formal concept analysis. In: Trappl, R. (ed.) Proc. of the 20th European Meeting on Cybernetics and Systems Research 2010, Vienna, Austria, pp. 32–38 (2010), ISBN 3–85206–178–8
2. Belohlavek, R., Sklenář, V.: Formal concept analysis constrained by attribute-dependency formulas. In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 176–191. Springer, Heidelberg (2005)
3. Belohlavek, R., Vychodil, V.: Formal concept analysis with constraints by closure operators. In: Schärfe, H., Hitzler, P., Ohrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 131–143. Springer, Heidelberg (2006)
4. Belohlavek, R., Vychodil, V.: Formal concept analysis with background knowledge: attribute priorities. IEEE Trans. Systems, Man, and Cybernetics, Part C 39(4), 399–409 (2009), doi:10.1109/TSMCC.2008.2012168

5. Boulicaut, J.-F., Jeudy, B.: Constraint-based data mining. In: Maimon, O., Rokach, L. (eds.) The Data Mining and Knowledge Discovery Handbook, pp. 399–416. Springer, Heidelberg (2005)
6. Carpineto, C., Romano, G.: Concept Data Analysis. Theory and Applications. J. Wiley, Chichester (2004)
7. Cellier, P., Ferré, S., Ridoux, O., Ducassé, M.: A parameterized algorithm to explore formal contexts with a taxonomy. Int. J. Found. Comput. Sci. 2, 319–343 (2008)
8. Dias, S.M., Vieira, N.J.: Reducing the size of concept lattices: The JBOS Approach. In: Proc. CLA 2010. CEUR WS, vol. 672, pp. 80–91 (2010), ISBN 978-84-614-4027-6, ISSN 1613–0073
9. Ganter, B.: Attribute exploration with background knowledge. Theor. Comput. Sci. 217, 215–233 (1999)
10. Ganter, B., Wille, R.: Formal Concept Analysis. Mathematical Foundations. Springer, Berlin (1999)
11. Grabisch, M., Marichal, J.-L., Mesiar, R., Pap, E.: Aggregation Functions. Cambridge University Press, Cambridge (2009)
12. Kwuida, L., Missaoui, R., Ben Amor, B., Boumedjout, L., Vaillancourt, J.: Restrictions on concept lattices for pattern management. In: Proc. CLA 2010. CEUR WS, vol. 672, pp. 235–246 (2010), ISBN 978-84-614-4027-6 , ISSN 1613–0073
13. Roberts, F.S.: Measurement Theory. Cambridge University Press, Cambridge (1985)
14. Xie, Z.: Φ-generalized concept lattice models: From power concepts to formal concepts, and then to robust concepts. In: Proc. CLA 2006, pp. 219–230 (2006)

Some Complexity Results about Essential Closed Sets

Felix Distel

Theoretical Computer Science, TU Dresden, Germany
felix@tcs.inf.tu-dresden.de

Abstract. We examine the enumeration problem for essential closed sets of a formal context. Essential closed sets are sets that can be written as the closure of a pseudo-intent. The results for enumeration of essential closed sets are similar to existing results for pseudo-intents, albeit some differences exist. For example, while it is possible to compute the lectically first pseudo-intent in polynomial time, we show that it is not possible to compute the lectically first essential closed set in polynomial time unless P = NP. This also proves that essential closed sets cannot be enumerated in the lectic order with polynomial delay unless P = NP. We also look at minimal essential closed sets and show that they cannot be enumerated in output polynomial time unless P = NP.

1 Introduction

The analysis of dependencies between attributes, so-called *implications*, is an important area of research within Formal Concept Analysis (FCA). Already in [7] it has been shown how a complete set of implications with minimal cardinality can be obtained from a formal context. This set is now commonly known as the *Duquenne-Guigues-Base* of a context. Since its discovery many results and algorithms in FCA, such as Attribute Exploration, have made use of the Duquenne-Guigues-Base.

Not surprisingly, a lot of effort has been directed at finding efficient algorithms to compute it. One of the earliest, and probably most well-known algorithms is *Next-Closure-Algorithm* [5]. It produces all concept intents and all pseudo-intents of a given formal context in a lexicographic order (called the *lectic order*). During the last decade, newer algorithms have been developed [9,11].

It is known that the Duquenne-Guigues base cannot be computed in polynomial time in the size of the input, since the base itself can be exponentially large in the size of the input [8]. This leaves the question whether it can be enumerated in output-polynomial time. Until now, no output-polynomial algorithm has been found, and it is also not known whether such an algorithm exists. Recently, a lot of progress has been made with respect to this question. It has been shown that the implications from the Duquenne-Guigues Base cannot be enumerated in output-polynomial time unless the transversal hypergraph problem (cf. [4]) is in P [12,13]. In [2] a connection between the boolean satisfiability problem (SAT) and enumeration problems from FCA has been established.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 81–92, 2011.

In particular, it has been shown using a reduction from SAT that the Duquenne-Guigues-Base cannot be enumerated with polynomial delay in the lectic order unless P = NP. A reduction from SAT has also been used in [1] to show that the problem of verifying whether a given set of attributes is a *pseudo-intent*, i. e. whether it occurs as the left-hand side of an implication in the Duquenne-Guigues Base, is coNP-complete. In the same paper it is also shown that pseudo-intents cannot be enumerated in the reverse lectical order with polynomial delay. Other works related to enumeration algorithms for pseudo-intents include [6] where optimizations based on hidden dependencies within the Duquenne-Guigues Base are considered. In [10] it is shown that the problem of counting pseudo-intents is #P-hard.

Previous work has mainly considered the pseudo-intents, i. e. the left-hand sides of the implications. In this paper we look at the right-hand sides, which are commonly called *essential closed sets*. In [1] it is shown that verifying whether a given set of attributes of a context is a pseudo-intent is as hard as verifying whether it is an essential closed set, i. e. it is coNP-complete. Unfortunately, a similar connection cannot be easily obtained for the decision problems considered in [2]. We therefore present yet another reduction from SAT which yields several complexity results about essential closed sets. Most of these results are similar to the ones for pseudo-intents. The main part of this paper is a reduction from SAT which proves that the problem of verifying whether a given set of attributes contains an essential closed set is NP-complete (Section 3). In Section 4 several other results are obtained using the same reduction. In particular, it is shown that the lectically first essential closed set cannot be computed in polynomial time unless P = NP, that essential closed sets cannot be enumerated in the lectic order with polynomial delay unless P = NP, and that minimal essential closed sets cannot be enumerated in output polynomial time unless P = NP.

2 Preliminaries

A *formal context* is a tuple (G, M, I) where G and M are finite sets and $I \subseteq G \times M$ is a binary relation. The elements of G are called *objects* and elements of M are called *attributes*. For a set of objects $A \subseteq G$ its *derivation* A' is defined as

$$A' = \{m \in M \mid \forall g \in A : gIm\}.$$

Analogously, for a set $B \subseteq M$ its *derivation* B' is defined as

$$B' = \{g \in G \mid \forall m \in B : gIm\}.$$

Applying the two derivation operators successively yields the closure operators $\cdot''$. Whenever we speak of a *closed set* in this work, we mean a set of attributes $B \subseteq M$ that is closed with respect to $\cdot''$, i. e. that satisfies $B'' = B$. Sets of attributes that can be written as $\{g\}'$ for some $g \in G$ are called *object intents*. The following result is common knowledge in FCA.

Proposition 1. *A set of attributes $B \subseteq M$ is closed if and only if it can be written as an intersection of object intents, i. e. there is a set $A \subseteq G$ such that*

$$B = \bigcap_{g \in A} \{g\}'.$$

A relevant research area in FCA are dependencies between sets of attributes. The simplest form of such a dependency is an implication $A \to B$, $A, B \subseteq M$. A set of attributes $D \subseteq M$ respects $A \to B$ if $A \not\subseteq D$ or $B \subseteq D$. $A \to B$ holds in the context (G, M, I) if all object intents respect $A \to B$.

Let $\mathcal{L}$ be a set of implications. We say that $A \to B$ follows semantically from $\mathcal{L}$ if and only if each subset $D \subseteq M$ that respects all implications from $\mathcal{L}$ also respects $A \to B$. $\mathcal{L}$ is an implicational base for (G, M, I) if it is

- *sound*, i. e. all implications from $\mathcal{L}$ hold in (G, M, I), and
- *complete*, i. e. all implications that hold in (G, M, I) follow from $\mathcal{L}$.

In [7] a minimum cardinality base, which is called the *Duquenne-Guigues-Base*, has been introduced. The left-hand sides of the implications in the Duquenne-Guigues-Base are called *pseudo-intents*. $P \subseteq M$ is a *pseudo-intent of* $\mathbb{K}$ if P is not closed and $Q'' \subseteq P$ holds for every pseudo-intent Q that is a proper subset of P. The Duquenne-Guigues-Base consists of all implications $P \to P''$, where P is a pseudo-intent. A set $R \subseteq M$ is an *essential closed set (of* $\mathbb{K}$*)* if there is a pseudo-intent P of $\mathbb{K}$ satisfying $P'' = Q$. Hence, the essential closed sets of a context are exactly those sets that occur as the right-hand side of an implication in the Duquenne-Guigues-Base. The following result is also common knowledge from FCA.

Proposition 2. *Let $\mathbb{K}$ be a formal context and let $Q \subseteq M$ be a set of attributes. If Q is not closed then Q contains a pseudo-intent of $\mathbb{K}$.*

The most well-known algorithm for computing the Duquenne-Guigues-Base is Next-Closure. It computes the set of all closed sets and all pseudo-intents of a context $\mathbb{K}$ in a special order, called the *lectic order*. Let $<$ be a total order on the elements of M. Then we say that $A \subseteq M$ is *lectically smaller* than $B \subseteq M$ if the smallest element with respect to $<$ that distinguishes A and B is contained in B. Formally, we write

$$A < B :\Leftrightarrow \exists x \in B \setminus A : \forall y < x : (y \in A \Leftrightarrow y \in B).$$

Notice that the lectic order extends the subset order, i. e. $A \subsetneq B$ implies $A < B$.

3 Main Reduction

In this section we prove that the following auxiliary problem is NP-hard. All other results will be based on this reduction and can be found in Section 4.

*Problem 1 (Essential Closed Subset (*ECS*)). Input:* A formal context $\mathbb{K} = (G, M, I)$ and a set $B \subseteq M$.
Question: Does an essential closed set $Q \subseteq B$ of $\mathbb{K}$ exist?

We prove NP-hardness using a reduction from SAT.

*Problem 2 (*SAT*). Input:* A boolean CNF-formula $f(p_1,\dots,p_n) = C_1 \wedge \dots \wedge C_m$, where $C_i = (x_{i1} \vee \dots \vee x_{il_i})$ and $x_{ij} \in \{p_1,\dots,p_n\} \cup \{\neg p_1,\dots,\neg p_n\}$ for all $i \in \{1,\dots,m\}$ and all $j \in \{1,\dots,l_i\}$.
Question: Is f satisfiable?

SAT remains NP-complete if we impose the additional condition that for every literal $l \in \{p_1,\dots,p_n\} \cup \{\neg p_1,\dots,\neg p_n\}$ there is a clause C_i in which l does not occur (otherwise we could simply add a new variable p_{n+1} and a new clause $C_{m+1} = (p_{n+1})$ without changing satisfiability of the formula).

Let an instance of SAT, i. e. a formula $f(p_1,\dots,p_n) = C_1 \wedge \dots \wedge C_m$, where $C_i = (x_{i1} \vee \dots \vee x_{il_i})$ and $x_{ij} \in \{p_1,\dots,p_n\} \cup \{\neg p_1,\dots,\neg p_n\}$ for all $i \in \{1,\dots,m\}$ and all $j \in \{1,\dots,l_i\}$, be given. We construct an instance of ECS. We define

$$M = \{\alpha_1,\dots,\alpha_n\} \cup \{p_1,\dots,p_n\} \cup \{\neg p_1,\dots,\neg p_n\} \cup \{\beta\} \tag{1}$$

For every $r \in \{1,\dots,n\}$ we define sets T_r and F_r.

$$\begin{aligned} T_r &= M \setminus \{\neg p_r, \alpha_r\} \\ F_r &= M \setminus \{p_r, \alpha_r\} \end{aligned} \tag{2}$$

Finally, for every $i \in \{1,\dots,m\}$ we define a set

$$\begin{aligned} A_i = M &\setminus \{\beta\} \\ &\setminus \{p_r \mid r \in \{1,\dots,n\},\ \text{the positive literal } p_r \text{ occurs in } C_i\} \\ &\setminus \{\neg p_r \mid r \in \{1,\dots,n\},\ \text{the negative literal } \neg p_r \text{ occurs in } C_i\} \\ &\setminus \{\alpha_r \mid r \in \{1,\dots,n\},\ p_r \text{ or } \neg p_r \text{ occurs in } C_i\} \end{aligned} \tag{3}$$

We construct a context $\mathbb{K}_f = (G, M, I)$ whose attribute set M is defined as in (1), whose set of objects is

$$G = \{g_{A_1},\dots,g_{A_m}\} \cup \{g_{T_1},\dots,g_{T_n}\} \cup \{g_{F_1},\dots,g_{F_n}\} \cup \{g_{Q_1},\dots,g_{Q_n}\},$$

and whose incidence relation I is such that

$$\{g_{A_i}\}' = A_i, \quad \{g_{T_r}\}' = T_r, \quad \{g_{F_r}\}' = F_r, \quad \{g_{Q_r}\}' = \{\alpha_r, p_r, \neg p_r\}$$

for all $r \in \{1,\dots,n\}$ and all $i \in \{1,\dots,m\}$. This context is shown in Table 1.

Our eventual objective is to reduce SAT to ECS by proving that f is satisfiable if and only if there exists a subset of

$$B = M \setminus \{\alpha_1,\dots,\alpha_n\}$$

that is an essential closed set of $\mathbb{K}_f$. We need several technical results.

Let $\phi : \{p_1,\dots,p_n\} \to \{\texttt{true}, \texttt{false}\}$ be an assignment that assigns truth values to all variables. There is a natural correspondence between ϕ and a set of attributes S_ϕ. We define

$$S_\phi := \{p_r \mid \phi(p_r) = \texttt{true}\} \cup \{\neg p_r \mid \phi(p_r) = \texttt{false}\}. \tag{4}$$

The following result motivates our choice of the object intents $\{g_{A_i}\}' = A_i$, $i \in \{1,\dots,m\}$.

Table 1. Context $\mathbb{K}_f$

	α_1 … α_n	p_1 … p_n	$\neg p_1$ … $\neg p_n$	β
g_{A_1}	⋯	A_1	⋯	
⋮		⋮		
g_{A_m}	⋯	A_m	⋯	
g_{T_1}	⋯	T_1	⋯	
⋮		⋮		
g_{T_n}	⋯	T_n	⋯	
g_{F_1}	⋯	F_1	⋯	
⋮		⋮		
g_{F_n}	⋯	F_n	⋯	
g_{Q_1}	×	×	×	
⋮	⋱	⋱	⋱	
g_{Q_n}	×	×	×	

Lemma 1. *Let ϕ be an assignment of truth values. Then ϕ makes f true if and only if $S_\phi \not\subseteq A_i$ holds for all $i \in \{1, \ldots, m\}$.*

Proof. Since f is in conjunctive normal form, ϕ makes f true if and only if ϕ makes every clause C_i, $i \in \{1, \ldots, m\}$, of f true. For every $i \in \{1, \ldots, m\}$ the assignment ϕ makes the clause C_i true if and only if one of the literals in C_i evaluates to true, i. e.

- there is some p_r satisfying $\phi(p_r) = \texttt{true}$, where the positive literal p_r occurs in C_i, or
- there is some p_r satisfying $\phi(p_r) = \texttt{false}$, where the negative literal $\neg p_r$ occurs in C_i.

According to (3) and (4) this is equivalent to saying that

- there is some $p_r \in S_\phi$, where $p_r \notin A_i$, or
- there is some $\neg p_r \in S_\phi$, where $\neg p_r \notin A_i$.

This is equivalent to $S_\phi \not\subseteq A_i$. Thus we have shown that ϕ makes f true if and only if $S_\phi \not\subseteq A_i$ holds for all $i \in \{1, \ldots, m\}$.

The following proposition follows immediately from (2) and (4).

Proposition 3. *Let ϕ be an assignment of truth values and $X \subseteq S_\phi$ a set of attributes. Then*

$$X \cup \{\beta\} = \bigcap_{\neg p_r \notin X} T_r \cap \bigcap_{p_r \notin X} F_r \tag{5}$$

holds. Since for all $r \in \{1, \ldots, n\}$ the sets T_r and F_r are object intents this proves that $X \cup \{\beta\}$ is closed.

Proposition 4. *Let ϕ be an assignment of truth values and $X \subseteq S_\phi$ a set of attributes. X is closed if and only if there is some $i \in \{1, \dots, m\}$ such that $X \subseteq A_i$ holds.*

Proof. ($\Leftarrow$) We already know from Proposition 3 that $X \cup \{\beta\} = \bigcap_{\neg p_r \notin X} T_r \cap \bigcap_{p_r \notin X} F_r$ holds for X. Since $X \subseteq A_i$ and $\beta \notin A_i$ it follows that

$$X = A_i \cap \big(X \cup \{\beta\}\big) = A_i \cap \bigcap_{\neg p_r \notin X} T_r \cap \bigcap_{p_r \notin X} F_r.$$

Since T_r and F_r, $r \in \{1, \dots, n\}$, and A_i are object intents Proposition 1 proves that $X \cup \{\beta\}$ is closed.

($\Rightarrow$) The case where $X = \emptyset$ is trivial. Let $X = \{l\}$ be a singleton set. We have required that for every literal there is a clause in which it does not occur. Hence, there is a clause C_i in which l does not occur, and therefore $X = \{l\} \subseteq A_i$ holds. The case where X contains at least two elements remains. Since $X \subseteq S_\phi$ holds, X cannot contain $\{p_r, \neg p_r\}$ or $\{\alpha_r\}$ for any $r \in \{1, \dots, n\}$. We obtain that if X has at least two elements then it cannot be a subset of $\{\alpha_r, p_r, \neg p_r\} = \{g_{Q_r}\}'$ for any $r \in \{1, \dots, n\}$. Assume that $X \not\subseteq A_i = \{g_{A_i}\}'$ holds for all $i \in \{1, \dots, m\}$. Then the only object intents that contain X are of the form $\{g_{T_r}\}'$ or $\{g_{F_r}\}'$. All object intents of the form $\{g_{T_r}\}'$ or $\{g_{F_r}\}'$ contain β, which yields $\beta \in X''$. Because $\beta \notin S_\phi \supseteq X$ this is a contradiction to the fact that X is closed. Therefore, the assumption that $X \not\subseteq A_i$ holds for all $i \in \{1, \dots, m\}$ must be false, i. e. there must be some $i \in \{1, \dots, m\}$ satisfying $X \subseteq A_i$.

Lemma 2. *For every $r \in \{1, \dots, n\}$ it holds that*

$$\{\alpha_r\}'' = \{p_r, \neg p_r\}'' = \{\alpha_r, p_r, \neg p_r\}.$$

Proof. We have defined $\mathbb{K}_f$ in such a way that every object intent that contains $\{p_r, \neg p_r\}$ also contains $\{\alpha_r\}$. Conversely, every object intent that contains $\{\alpha_r\}$ also contains $\{p_r, \neg p_r\}$. This proves $\{\alpha_r, p_r, \neg p_r\} \subseteq \{p_r, \neg p_r\}''$ and $\{\alpha_r, p_r, \neg p_r\} \subseteq \{\alpha_r\}''$. On the other hand, we know that $\{\alpha_r, p_r, \neg p_r\} = \{g_{Q_r}\}'$ is closed since it is an object intent. This yields $\{\alpha_r, p_r, \neg p_r\} = \{p_r, \neg p_r\}'' = \{\alpha_r\}''$.

Theorem 1. *Define $B = \{p_1, \dots, p_n\} \cup \{\neg p_1, \dots, \neg p_n\} \cup \{\beta\}$. There is an essential closed set $Q \subseteq B$ if and only if f is satisfiable.*

Proof. ($\Rightarrow$) Assume that Q contains both p_r and $\neg p_r$ for some $r \in \{1, \dots, n\}$. Lemma 2 yields $\alpha_r \in Q''$. This contradicts the fact that Q is a closed subset of B. Therefore, the assumption that Q contains both p_r and $\neg p_r$ for some $r \in \{1, \dots, n\}$ must be false. Thus, Q must be a subset of $S_\phi \cup \{\beta\}$ for some assignment ϕ. Since Q is an essential closed set there must be a pseudo-intent $P \subseteq Q \subseteq S_\phi \cup \{\beta\}$. If P contains β then Proposition 3 yields that P is closed. This contradicts the fact that P is a pseudo-intent. Hence, P cannot contain β, i. e. $P \subseteq S_\phi$ holds. Since P is a pseudo-intent and therefore not closed we

obtain from Proposition 4 that there is no $i \in \{1, \ldots, m\}$ such that $P \subseteq A_i$. $P \subseteq S_\phi$ yields that there is no $i \in \{1, \ldots, m\}$ such that $S_\phi \subseteq A_i$. It follows from Lemma 1 that ϕ makes f true.

($\Leftarrow$) Let ϕ be an assignment that makes f true. Lemma 1 implies $S_\phi \not\subseteq A_i$ for all $i \in \{1, \ldots, m\}$. Proposition 4 shows that S_ϕ is not closed. Let X be minimal among all subsets of S_ϕ that are not closed. Then in particular all subsets of X are closed. Since X is not closed, but all of its subsets are closed, X must be a pseudo-intent of $\mathbb{K}_f$. Proposition 3 states that $X \cup \{\beta\}$ is closed, and therefore $X'' = X \cup \{\beta\}$ holds. This shows that $X \cup \{\beta\}$ is an essential closed set. Since $X \cup \{\beta\}$ is also a subset of B this proves the initial claim.

Corollary 1. ECS *is* NP*-hard.*

Proof. Every boolean formula f can be converted into an instance of ECS, namely the context $\mathbb{K}_f$ and the set $B = \{p_1, \ldots, p_n\} \cup \{\neg p_1, \ldots, \neg p_n\} \cup \{\beta\}$, in polynomial time. Theorem 1 states that f is a "Yes"-instance of SAT if and only if $\mathbb{K}_f$ and B are a "Yes"-instance of ECS.

We have thus shown that the problem of deciding whether a given set of attributes B in a context $\mathbb{K}$ contains an essential closed set, is NP-hard. Surprisingly, the problem becomes easier if we require B to be closed. If all subsets of B are closed then B cannot contain a pseudo-intent, and thus it does not contain an essential closed set. On the other hand, if B contains a set S that is not closed, then there must be a pseudo-intent $P \subseteq S$ because of Proposition 2. We obtain $P'' \subseteq S'' \subseteq B'' = B$. Therefore B contains the essential closed set P''. This proves that checking whether a closed set B contains an essential closed set is equivalent to checking whether all subsets of B are closed. It is well known that the latter can be done in polynomial time.

4 Further Results

Let $\mathbb{K} = (G, M, I)$ be a formal context. We call a set Q is a *minimal essential closed set (of* $\mathbb{K}$*)* if Q is minimal with respect to set inclusion among all essential closed sets of $\mathbb{K}$. It is known from [1] that the problem of deciding whether a given set of attributes is an essential closed set is coNP-hard. We first show that the problem becomes easier for minimal essential closed sets: it is possible to decide in polynomial time whether a given set is a minimal essential closed set. This result is required for later proofs.

Proposition 5. *Q is a minimal essential closed set if and only if*

- *Q is closed, and*
- *not every subset of Q is closed, and*
- *every closed set $R \subsetneq Q$ satisfies*

$$\forall S \subseteq R : S'' = S. \tag{6}$$

Proof. ($\Rightarrow$) As an essential closed set, Q is obviously closed. As an essential closed set, Q must contain a pseudo-intent P_1, which is not closed. Hence, not all subsets of Q are closed. Assume that there is a strict subset $R \subsetneq Q$ that is closed and a set $S \subseteq R$ that is not closed. By Proposition 2 S contains a pseudo-intent $P_2 \subseteq S \subseteq R$. Since R is closed it follows that $P_2'' \subseteq R \subsetneq Q$. Hence, P_2'' is an essential closed set and a strict subset of Q, which contradicts minimality of Q. Thus the assumption that such a set S exists must be false.

($\Leftarrow$) Since not all subsets of Q are closed there must be a pseudo-intent $P \subseteq Q$ by Proposition 2. Since Q is closed we obtain $P'' \subseteq Q$. P'' cannot be a strict subset of Q, because otherwise (6) would imply that P is closed. Therefore, $P'' = Q$ holds, and thus Q is an essential closed set. No strict subset of Q can be an essential closed set because of (6). Thus Q is a minimal essential closed set.

Notice that in order to decide whether a given set Q satisfies (6) for all closed sets $R \subsetneq Q$ it suffices to check whether (6) holds for all sets R that are maximal with respect to set inclusion among the closed strict subsets of Q. If Q is itself closed then these are of the form $Q \cap \{g\}'$, where $g \in G$ and $Q \not\subseteq \{g\}'$. Hence, it suffices to check (6) for at most $|G|$ strict subsets of Q. It has been established in previous works [2] that one can decide in polynomial time whether all subsets of a given set of attributes are closed. Hence, all conditions from Propostion 5 can be tested in polynomial time.

Corollary 2. *Let $\mathbb{K}$ be a formal context and $Q \subseteq M$ a set of attributes. It is possible to decide in time polynomial in the size of the context $\mathbb{K}$ and the size of Q whether Q is a minimal essential closed set.*

This gives us the containment result corresponding to the hardness result from Corollary 1. Clearly, for a formal context $\mathbb{K} = (G, M, I)$ and a set $B \subseteq M$ there is an essential closed set $Q \subseteq B$ if and only if there is a minimal essential closed set $R \subseteq B$. In order to decide in non-deterministic polynomial time whether B contains an essential closed set we can non-deterministically guess a subset of B and decide using Corollary 2 whether it is a minimal essential closed set. This proves that ECS is contained in NP. Together with the previous hardness result we obtain NP-completeness.

Theorem 2. ECS *is* NP*-complete.*

We want to take a closer look at the enumeration problem for essential closed sets. But first we consider the following decision problem.

*Problem 3 (Lectically Smaller Essential Closed Set (*LS-ECS*)). Input:* A formal context $\mathbb{K} = (G, M, I)$ and a set $B \subseteq M$.
Question: Is there an essential closed set Q of $\mathbb{K}$ which is lectically smaller that B?

Theorem 3. LS-ECS *is* NP*-complete.*

Proof. Containment: Since the lectic order extends the subset order there is an essential closed set that is lectically smaller than B if and only if there is a minimal essential closed set that is lectically smaller than B. This can be verified by non-deterministically guessing a subset of B and checking in polynomial time whether it is a minimal essential closed set.

Hardness: Given an instance f of SAT we can construct an instance of LS-ECS consisting of the context $\mathbb{K}_f$ and the set $B = \{p_1, \ldots, p_n\} \cup \{\neg p_1, \ldots, \neg p_n\} \cup \{\beta\}$ using the same reduction as in Section 3. We define the order on the set of attributes as follows

$$\alpha_1 < \cdots < \alpha_n < p_1 < \cdots < p_n < \neg p_1 < \cdots < \neg p_n < \beta.$$

Then the sets that are lectically smaller than B are exactly the subsets of B. The correctness of the reduction therefore follows from Theorem 1.

This result has consequences for the problem of enumeration of essential closed sets in the lectic order. If it were possible to compute the lectically first essential closed set of a context $\mathbb{K}$ in polynomial time then we could decide LS-ECS in polynomial time as follows. We would simply compute the lectically first essential closed set of $\mathbb{K}$ and check whether it is lectically smaller than B. Because of Theorem 3 it is not possible to decide LS-ECS in polynomial time unless P = NP.

Corollary 3. *Let $\mathbb{K}$ be a formal context. It is not possible to compute the lectically first essential closed set of $\mathbb{K}$ in polynomial time unless* P = NP.

In this respect, the computational behaviour of essential closed sets is worse than that of pseudo-intents, since the lectically first pseudo-intent can be computed in polynomial time [3]. Because not even the lectically first essential closed set can be computed in polynomial time it is obviously not possible to enumerate essential closed sets in the lectic order with polynomial delay.

Corollary 4. *It is not possible to enumerate essential closed sets in the lectic order with polynomial delay.*

Finally, we consider the problem of enumeration of minimal essential closed sets.

*Problem 4 (All Minimal Essential Closed Sets (*ALL-MECS*)). Input:* A formal context $\mathbb{K} = (G, M, I)$ and sets $Q_1, \ldots, Q_k \subseteq M$.
Question: Are $Q_1, \ldots, Q_k$ all minimal essential closed sets of $\mathbb{K}$?

Lemma 3. *Let f be a boolean CNF-formula and $\mathbb{K}_f$ the formal context constructed as in Section 3. Then $Q_1 = \{\alpha_1, p_1, \neg p_1\}$, $\ldots$, $Q_n = \{\alpha_n, p_n, \neg p_n\}$ are all the minimal essential closed sets of $\mathbb{K}_f$ if and only if there is no essential closed set $Q \subseteq B = \{p_1, \ldots, p_n\} \cup \{\neg p_1, \ldots, \neg p_n\} \cup \{\beta\}$.*

Proof. ($\Rightarrow$) Assume that there is an essential closed set $Q \subseteq B$. Then Q contains a minimal essential closed set R. For all $r \in \{1, \ldots, n\}$ since B does not contain α_r the set R cannot contain α_r, either. Thus $R \neq Q_r$ holds for all $r \in \{1, \ldots, n\}$, which contradicts the fact that $Q_1, \ldots, Q_n$ are all the minimal essential closed sets of $\mathbb{K}$. Hence, the assumption must be false, i. e. an essential closed set $Q \subseteq B$ cannot exist.

($\Rightarrow$) By Lemma 2 every closed set that contains α_r for some $r \in \{1, \ldots, n\}$ must also contain $\{p_r, \neg p_r\}$. Therefore, $Q_r = \{\alpha_r, p_r, \neg p_r\}$ is the only minimal essential closed set of $\mathbb{K}_f$ that contains α_r. Thus, every essential closed set that is different from $Q_1, \ldots, Q_n$ must be a subset of B. The hypothesis states that such a set does not exist. Hence, $Q_1, \ldots, Q_n$ are all the minimal essential closed sets of $\mathbb{K}$.

Theorem 4. ALL-MECS *is* coNP*-complete.*

Proof. To prove hardness using a reduction from SAT from a given formula f we construct a context $\mathcal{K}_f$ as in Section 3 and sets $Q_1, \ldots, Q_n$ as in Lemma 3. Lemma 3 and Theorem 1 show that $Q_1, \ldots, Q_n$ are all the minimal essential closed sets of $\mathbb{K}$ if and only if f is not satisfiable. This proves that ALL-MECS is coNP-hard. Containment can be shown using Corollary 2: Given an instance of ALL-MECS consisting of a context $\mathbb{K} = (G, M, I)$ and sets $Q_1, \ldots, Q_n$ we can verify in polynomial time using Corollary 2 that all sets $Q_1, \ldots, Q_n$ are minimal essential closed sets. Subsequently, we non-deterministically guess a set $S \subseteq M$ and check in polynomial time whether it is a minimal essential closed set that is different from $Q_1, \ldots, Q_n$.

If there were an algorithm $\mathcal{A}$ that enumerates the minimal essential closed sets of a context in output polynomial time, then we could construct an algorithm $\mathcal{A}'$ that decides ALL-MECS as follows: Since $\mathcal{A}$ can enumerate the minimal essential closed sets of a context $\mathbb{K}$ in output polynomial time there must be a polynomial $p(k, n)$ that serves as an upper bound for the runtime of $\mathcal{A}$, where k is the size of the input context $\mathbb{K}$ and n is the number of minimal essential closed sets of $\mathbb{K}$. To decide ALL-MECS for a context $\mathbb{K}$ and sets $Q_1, \ldots, Q_n$ we let $\mathcal{A}$ run on $\mathbb{K}$ and stop it after time $p(|\mathbb{K}|, n)$. Then we compare its output (if any) to $Q_1, \ldots, Q_n$. $Q_1, \ldots, Q_n$ are not all the minimal essential closed sets of $\mathbb{K}$ iff the outputs differ or $\mathcal{A}$ does not terminate within $p(|\mathbb{K}|, n)$ steps. Since ALL-MECS cannot be decided in polynomial time unless P = NP such an algorithm cannot exist unless P = NP.

Corollary 5. *Minimal essential closed sets cannot be enumerated in output polynomial time unless* P = NP*.*

5 Conclusion

Using a new reduction from SAT we have shown several complexity results about essential closed sets. Most of these results closely resemble those for

pseudo-intents. Essential closed sets cannot be enumerated in the lectic order with polynomial delay unless P = NP, and minimal essential closed sets cannot be enumerated in output polynomial time unless P = NP. The same holds for pseudo-intents [2].

Essential closed sets differ from pseudo-intents computationally with respect to the following problems. For an arbitrary set of attributes it is NP-hard to verify whether it contains an essential closed set. By contrast, it is easy to check whether a given set of attributes contains a pseudo-intent, because this simply means checking for closedness. It is impossible to compute the lectically first essential closed set unless P = NP. The lectically first pseudo-intent can be computed in polynomial time [3].

These results are, of course, only a minor contribution to the question whether the Duquenne-Guigues Base can be enumerated efficiently. This important question remains open and should be part of future work.

References

1. Babin, M., Kuznetsov, S.: Recognizing pseudo-intents is coNP-complete. In: Kryszkiewicz, M., Obiedkov, S. (eds.) Proc. of the 7th Int. Conf. on Concept Lattices and Their Applications (CLA 2010). CEUR Workshop Proceedings, vol. 672 (2010)
2. Distel, F.: Hardness of enumerating pseudo-intents in the lectic order. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA 2010. LNCS, vol. 5986, pp. 124–137. Springer, Heidelberg (2010)
3. Distel, F., Sertkaya, B.: On the complexity of enumerating pseudo-intents. Discrete Applied Mathematics (2011) (to appear)
4. Eiter, T., Gottlob, G.: Hypergraph transversal computation and related problems. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 549–564. Springer, Heidelberg (2002)
5. Ganter, B.: Two basic algorithms in concept analysis. Preprint 831, Fachbereich Mathematik, TU Darmstadt, Darmstadt, Germany (1984)
6. Gély, A., Medina, R., Nourine, L., Renaud, Y.: Uncovering and reducing hidden combinatorics in guigues-duquenne bases. In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 235–248. Springer, Heidelberg (2005)
7. Guigues, J.-L., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Math. Sci. Humaines 95, 5–18 (1986)
8. Kuznetsov, S.O.: On the intractability of computing the Duquenne-Guigues base. Journal of Universal Computer Science 10(8), 927–933 (2004)
9. Kuznetsov, S.O., Obiedkov, S.: Algorithms for the construction of concept lattices and their diagram graphs. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 289–300. Springer, Heidelberg (2001)
10. Kuznetsov, S.O., Obiedkov, S.: Counting Pseudo-intents and #P-completeness. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3874, pp. 306–308. Springer, Heidelberg (2006)

11. Obiedkov, S., Duquenne, V.: Attribute-incremental construction of the canonical implication basis. Annals of Mathematics and Artificial Intelligence 49(1-4), 77–99 (2007)
12. Sertkaya, B.: Some computational problems related to pseudo-intents. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS, vol. 5548, pp. 130–145. Springer, Heidelberg (2009)
13. Sertkaya, B.: Towards the complexity of recognizing pseudo-intents. In: Rudolph, S., Dau, F., Kuznetsov, S.O. (eds.) ICCS 2009. LNCS, vol. 5662, pp. 284–292. Springer, Heidelberg (2009)

A Context-Based Description of the Doubly Founded Concept Lattices in the Variety Generated by M_3

Stephan Doerfel[1,2]

[1] Knowledge & Data Engineering Group,
Department of Mathematics and Computer Science,
University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
doerfel@cs.uni-kassel.de
http://www.kde.cs.uni-kassel.de
[2] Department of Mathematics, Institute of Algebra,
Technical University of Dresden, 01062 Dresden, Germany

Abstract. In universal algebra and in lattice theory the notion of varieties is very prominent, since varieties describe the classes of all algebras (or of all lattices) modeling a given set of equations. While a comprehensive translation of that notion to a similar notion of varieties of complete lattices – and thus to Formal Concept Analysis – has not yet been accomplished, some characterizations of the doubly founded complete lattices of some special varieties (e.g. the variety of modular or that of distributive lattices) have been discovered. In this paper we use the well-known arrow relations to give a characterization of the formal contexts of doubly founded concept lattices in the variety that is generated by M_3 – the smallest modular, non-distributive lattice variety.

Keywords: variety, arrow-relations, modularity, projectivity, M_3, subdirect product.

1 Introduction

In lattice theory and more generally in universal algebra, varieties play a very important role. Given a set Σ of equations, the class of all algebras that model these equations (i. e. each of these equations holds in all these algebras) is denoted by $\mathrm{Mod}(\Sigma)$ and called a *variety*. Since generated by equations, in literature varieties are often called equational classes (e. g. in [5]). Although some of the results we discuss are true for all kinds of universal algebras, for the purpose of this paper it will suffice to speak only of lattice varieties. For a class of lattices $\mathbf{K}$ the set of all equations that hold for each lattice in $\mathbf{K}$ is denoted by $\mathrm{Eq}(\mathbf{K})$ and the class

$$\mathrm{Var}(\mathbf{K}) := \mathrm{Mod}(\mathrm{Eq}(\mathbf{K}))$$

is called the variety generated by $\mathbf{K}$. This is justified, since Mod and Eq form a Galois connection and thus $\mathrm{Mod}(\mathrm{Eq}(\mathbf{K}))$ is the smallest variety containing $\mathbf{K}$.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 93–106, 2011.

One can see that constructing varieties is a means to find related lattices to a set of given ones. I. e. $\mathbf{K} \mapsto \mathrm{Var}(\mathbf{K})$ is a closure operator yielding all lattices sharing all the properties that the lattices in $\mathbf{K}$ have in common. An important property of each variety is that it is closed under the formation of homomorphisms, sublattices and products. Moreover, varieties can be generated using these constructs via

$$\mathrm{Mod}(\mathrm{Eq}(\mathbf{K})) = \mathbf{HSP}(\mathbf{K}).$$

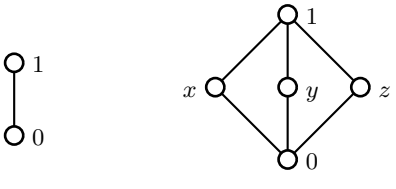
This fact is known as the **HSP**-*Theorem*. In this notion **H**, **S** and **P** are the closure operators that assign to a class of lattices its closure w.r.t. homomorphisms, sublattices and products. For details see e. g. [1] or [8].

Although many notions from lattice theory have been successfully adapted to the theory of complete lattices (which are no universal algebras), to the best of our knowledge the notion of varieties has not. It is easy to see that one can define operators $\mathrm{Mod_c}$ and $\mathrm{Eq_c}$ similar to the corresponding ones above only for complete lattices and polynomial complete (infinite) lattice equations (for a detailed explanation see [9], Section 2). $\mathrm{Mod_c}$ and $\mathrm{Eq_c}$ form a Galois connection and satisfy

$$\mathrm{Mod_c}(\mathrm{Eq_c}(\mathbf{K})) \subseteq \mathrm{Mod}(\mathrm{Eq}(\mathbf{K})) = \mathrm{Var}(\mathbf{K})$$

for a class of complete lattices $\mathbf{K}$. However, it is difficult to investigate these structures in general. One of the main problems is the non-existence of free complete lattices (free algebras play a key role in the study of varieties) – shown in [9], Section 3 (especially Theorem 1) – not even in the smallest non-trivial lattice variety (cf. [7]). Another problem arises while translating results from complete lattice theory to the context level. One often has to restrict the results to doubly founded lattices – in our case to the doubly founded complete members of a lattice variety. It should be noted, however, that each non-trivial lattice variety contains complete lattices that are not doubly founded.

For the variety of all modular lattices (in [4] or rephrased in [6], Theorem 42) and for the variety of all distributive lattices (in [3], Theorem 7.9, or rephrased in [6], Theorem 41) such context-based characterizations of the doubly founded complete members have been achieved. Both of these characterizations can be expressed using the arrow relations. Although these two varieties are probably the most interesting ones (due to the fact that the modular and the distributive law are the most important identities in lattice theory), also other varieties are



D_2 M_3

Fig. 1. Diagrams of the lattices D_2 and M_3

prominent, especially those in the lower part of the lattice of all lattice varieties (see Section 2). In this paper we focus on the variety generated by the lattice M_3 (see Figure 1), which is the smallest modular, non-distributive lattice variety.

The main result of this paper is the following theorem that characterizes the contexts of doubly founded members of the variety $\mathbf{M}_3 := \mathrm{Var}(\{M_3\})$ by the number of arrows in each row and each column of the context table.

Theorem 1. *For a clarified context (G, M, I) of a modular, doubly founded concept lattice the following conditions are equivalent:*

1. $\underline{\mathfrak{B}}(G, M, I) \in \mathbf{M}_3$.
2. $\forall g \in G : |\{m \in M \mid g \swarrow\!\!\nearrow m\}| \leq 2$.
3. $\forall m \in M : |\{g \in G \mid g \swarrow\!\!\nearrow m\}| \leq 2$.

The remainder of this paper is dedicated to the proof of that result. We therefore first recall several facts and definitions of Formal Concept Analysis and about modularity in the next section. Section 3 explains some useful facts about the lattice of modular varieties and about the position of $\mathbf{M}_3$ in that lattice. Section 4 contains the first part of the proof i. e. 1⟹2 and 1⟹3. In Section 5 we recall some results concerning projective lattices which will be applied in the last part of the proof of Theorem 1 presented in Section 6.

2 Arrows and Modularity

Since our characterization is performed by counting arrows in a context, this section begins with some facts about the arrow relations. Details can be found in [6], Chapter 1.2. The relations are defined by:

$$\begin{aligned} g \swarrow m &:\iff (g,m) \notin I \text{ and if } g' \subseteq h' \text{ and } g' \neq h', \text{ then } (h,m) \in I \\ g \nearrow m &:\iff (g,m) \notin I \text{ and if } m' \subseteq n' \text{ and } m' \neq n', \text{ then } (g,n) \in I \\ g \swarrow\!\!\nearrow m &:\iff g \swarrow m \text{ and } g \nearrow m \end{aligned}$$

where g is an object and m is an attribute of some context $\mathbb{K} = (G, M, I)$. The next Lemma presents a useful characterization of these relations. As usual, $\gamma g = (\{g\}'', \{g\}')$ denotes the object concept of an object g and $\mu m = (\{m\}', \{m\}'')$ the attribute concept of an attribute m. Further, we make use of the $*$ notation:

$$\begin{aligned} (A,B)_* &:= \sup\{(C,D) \in \underline{\mathfrak{B}}(\mathbb{K}) \mid (C,D) < (A,B)\} \quad \text{and} \\ (A,B)^* &:= \inf\{(C,D) \in \underline{\mathfrak{B}}(\mathbb{K}) \mid (C,D) > (A,B)\}. \end{aligned}$$

Lemma 1. *In (G, M, I) hold for each object $g \in G$ and each attribute $m \in M$*

$$\begin{aligned} g \swarrow m &\iff \gamma g \wedge \mu m = (\gamma g)_* \neq \gamma g \quad \textit{and} \\ g \nearrow m &\iff \gamma g \vee \mu m = (\mu m)^* \neq \mu m. \end{aligned}$$

The object g is irreducible, if and only if $(\gamma g)_* \neq \gamma g$. The attribute m is irreducible, if and only if $(\mu m)^* \neq \mu m$.

In Theorem 1 we require the concept lattices to be *modular* i. e. to satisfy the modular law. That is for a lattice L:

$$\forall x, y, z \in L : x \geq z \Longrightarrow x \wedge (y \vee z) = (x \wedge y) \vee z.$$

In order to express the requirements of Theorem 1 completely on the context level it could be rephrased using the following characterization of modularity for a doubly founded concept lattice $\underline{\mathfrak{B}}(G, M, I)$ (cf. [6], Theorem 42):

- From $g \swarrow m$, $g \swarrow n$, hIm and $h \not\mathrel{I} n$ follows that an attribute p exists with $h \not\mathrel{I} p$, gIp and $m' \cap n' \subseteq p'$, and
- from $g \nearrow m$, $h \nearrow m$, gIn and $h \not\mathrel{I} n$ follows that an object q exists with $q \not\mathrel{I} n$, qIm and $g' \cap h' \subseteq q'$.

In the next sections we will exploit two properties of modular lattices. The first is a well-known fact about the intervals and often referred to as the Isomorphism Theorem of modular lattices (cf. [8], Chapter IV.1, Theorem 2).

Theorem 2. *Let L be a modular lattice and let $a, b \in L$. Then*

$$\phi_b : [a, a \vee b] \to [a \wedge b, b],\ x \mapsto x \wedge b$$

is an isomorphism between the two intervals. The inverse isomorphism is

$$\psi_a : [a \wedge b, b] \to [a, a \vee b],\ y \mapsto y \vee a.$$

The other property concerns the arrows in contexts of modular lattices. A similar result for reduced contexts is stated in [6], Chapter 6.2.

Lemma 2. *In a context (G, M, I) of a modular concept lattice from $g \nearrow m$ or $g \swarrow m$ always follows $g \nearrow m$ for any irreducible object $g \in G$ and any irreducible attribute $m \in M$.*

Proof. Let $g \swarrow m$. By Lemma 1 we have $\gamma g \wedge \mu m = (\gamma g)_* \neq \gamma g$. According to the Isomorphism Theorem the intervals $[\gamma g \wedge \mu m, \gamma g]$ and $[\mu m, \mu m \vee \gamma g]$ are isomorphic. Thus, the neighborhood relation of $(\gamma g)_*$ and γg implies that $\mu m \vee \gamma g$ is an upper neighbor of μm. Since μm is infimum-irreducible, it can only have one upper neigbor. We obtain $\mu m \neq \mu m \vee \gamma g = (\mu m)^*$ and by Lemma 1 $g \nearrow m$. The second implication follows dually.

Apart from being modular we require the lattices we characterize to be doubly founded. A complete lattice L is called *doubly founded*, if for any two elements $x, y \in L$ there always are elements $s, t \in L$ with s being minimal w.r.t. $s \leq y$ and $s \not\leq x$ and t being maximal w.r.t. $t \geq x$ and $t \not\geq y$ (cf. [6], Definition 26). Note that for example every finite lattice is a doubly founded complete lattice. It is also possible to define doubly founded contexts. In fact, the context of a doubly founded lattice is always doubly founded. The definition can be found in [6], but the details are not important in this paper. In the proof of Theorem 1 we will employ the following lemma about products of doubly founded lattices.

Lemma 3. *A product of doubly founded lattices it doubly founded itself.*

Proof. Let I be an index set, L_i $(i \in I)$ be doubly founded complete lattices and $L = \prod_{i \in I} L_i$. Further, let $x = (x_i \mid i \in I)$ and $y = (y_i \mid i \in I)$ be two arbitrary elements of L satisfying $x < y$. From the latter follows that there exists an index $j \in I$ such that $x_j < y_j$ holds. Then, there is an element $\bar{s}_j \in L_j$ minimal w.r.t. $\bar{s}_j \leq y_j$ and $\bar{s}_j \not\leq x_j$. We set $s := (s_i \mid i \in I)$ with $s_j = \bar{s}_j$ and $s_i = 0$ for $i \neq j$. This yields $s \leq y$ and $s \not\leq x$, since clearly $s_j \not\leq x_j$. Now, for any element $t = (t_i \mid i \in I) \in L$ that is smaller than s we have $t_j < \bar{s}_j$ and $t_i = 0$ for $j \neq i$. But due to the construction of s_j from $t_j < \bar{s}_j$ follows $t_j \leq x_j$ and altogether $t \leq x$. Therefore, s has the claimed minimality property. Dually, one can construct a maximal element $t \in V$ with $t \geq x$ and $t \not\geq y$.

3 Varieties of Modular Lattices

The intersection of a set of varieties is again a variety and thus varieties form a lattice (ordered by inclusion). The modular varieties form a sublattice of the lattice of all varieties. The largest member of the lattice of modular varieties is of course the variety of all modular lattices. Trivially, the smallest variety in that lattice consists of all lattices with only one element (we do not count empty varieties here). The only upper neighbor of that variety is the distributive lattice variety, i. e. $\mathrm{Var}(\{D_2\})$. $\mathrm{Var}(\{D_2\})$ again has only one (modular) upper neighbor – the variety $\mathbf{M}_3$. A nice visualization of this situation is given by Figure 1 in [8], Chapter V.2. The next two results are both from Jónsson (cf. [11], Theorem 1 and Corollary 9). The first gives insight into varieties of modular lattices (such as $\mathbf{M}_3$), while the second one describes the upper neighbors of $\mathbf{M}_3$ in the lattice of all modular lattice varieties.

Theorem 3. *For any variety* $\mathbf{V}$ *of modular lattices the following conditions are equivalent:*

1. $M_{3,3} \notin \mathbf{V}$.
2. *Every member of* $\mathbf{V}$ *is isomorphic to a subdirect product of lattices of length two or less.*
3. *The inclusion* $a \wedge (b \vee (c \wedge d) \wedge (c \vee d)) \leq b \vee (a \wedge c) \vee (a \wedge d)$ *holds in* $\mathbf{V}$.

Corollary 1. *In the lattice of all varieties of modular lattices,* $\mathbf{M}_3$ *is covered by precisely two classes,* $\mathbf{M}_4 := \mathrm{Var}(\{M_4\})$ *and* $\mathbf{M}_{3,3} := \mathrm{Var}(\{M_{3,3}\})$*, and every class that properly contains* $\mathbf{M}_3$ *contains also* $\mathbf{M}_4$ *or* $\mathbf{M}_{3,3}$.

The diagrams of the lattices in the corollary are depicted in Figure 2. The following theorem is again a result from Jónsson (cf. [10], Corollary 3.4). It allows for a description of the members of the variety $\mathbf{M}_3$ as certain subdirect products. We will make use of this in Proposition 1.

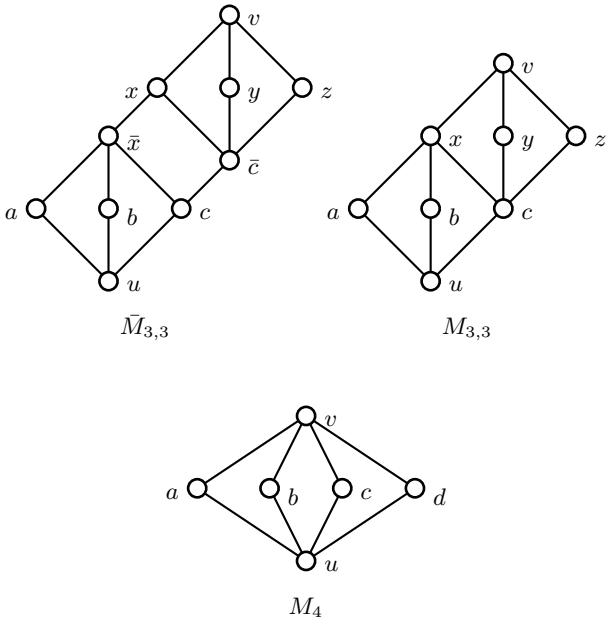


Fig. 2. Diagrams of the lattices $\bar{M}_{3,3}$, $M_{3,3}$ and M_4

Theorem 4. *If* $\mathbf{K}$ *is a finite set of finite algebras and if the lattice of all congruence relations over each algebra* $A \in \mathrm{Var}(\mathbf{K})$ *is distributive, then every subdirectly irreducible member of* $\mathrm{Var}(\mathbf{K})$ *belongs to* $\mathbf{HS}(\mathbf{K})$*, and hence*

$$\mathrm{Var}(\mathbf{K}) = \mathbf{IP}_S\mathbf{HS}(\mathbf{K}).$$

The operator $\mathbf{P}_S$ is the closure operator w.r.t. subdirect products and $\mathbf{I}$ is the closure operator w.r.t. isomorphic lattices. It is well known that the lattice of all congruence relations over any lattice is distributive (e. g. cf. [8], Chapter II.3, Theorem 11). From the above theorem therefore follows that every member of $\mathbf{M}_3$ is isomorphic to a subdirect product of lattices in $\mathbf{HS}(\{M_3\})$. Since further (according to Birkhoff's Theorem cf. [1], Chapter VIII.8, Theorem 15) each lattice is isomorphic to the subdirect product of subdirectly irreducible lattices, it is sufficient to consider only the subdirectly irreducible lattices in $\mathbf{HS}(\{M_3\})$, i. e. the lattices isomorphic to D_2 or to M_3 and lattices with only one element. However, the latter are not relevant for the construction of subdirect products that themselves have more than one element. We infer the following proposition.

Proposition 1. *Each lattice* $L \in \mathbf{M}_3$ *that has more than one element is isomorphic to a subdirect product of factors isomorphic to* D_2 *or to* M_3*.*

Diagrams of the two lattices D_2 and M_3 are shown in Figure 1. We are now ready to prove the first part of the claimed characterization.

4 Characterization Part I

Proposition 2. *If (G, M, I) is a clarified context of a complete lattice in the variety $\mathbf{M}_3$, then for every object $g \in G$ there are at most two distinct attributes $m, n \in M$, such that $g \not\nearrow m$ and $g \not\nearrow n$ hold. Dually, for every attribute $m \in M$ there are at most two distinct objects $g, h \in G$, such that $g \not\nearrow m$ and $h \not\nearrow m$ hold.*

Proof. Let L be the concept lattice of the context (G, M, I). We will prove the first part of the proposition indirectly and assume that there are three pairwise distinct attributes $m, n, p \in M$ with $g \not\nearrow m$, $g \not\nearrow n$ and $g \not\nearrow p$ for some object $g \in G$.

The set $X := \{\mu m, \mu n, \mu p, \gamma g\}$ of concepts of L generates a sublattice S in L. Because varieties are closed under the formation of sublattices, it follows from $L \in \mathbf{M}_3$ that S is also a member of $\mathbf{M}_3$. Therefore, by Proposition 1 there exist lattices L_t (for some fitting index set T) with L_t being isomorphic to either D_2 or to M_3 for each $t \in T$ – such that S is isomorphic to a subdirect product of the lattices L_t. For the sake of convenience we will assume that each of the factors is not only isomorphic but equal to either D_2 or to M_3 with the elements named $0, 1$ and $0, x, y, z, 1$ like in the diagrams in Figure 1.

We will now use this representation as subdirect product to produce a contradiction to our assumption by eliminating all possible combinations of such factors L_t. For each $t \in T$ let $\pi_t : S \to L_t,\ s \mapsto \pi_t s =: s_t$ denote the corresponding projection homomorphism. Thus, each element $s \in S$ is determined by the tuple $(s_t)_{t \in T}$. We can assume that there are no redundant factors in the subdirect product. That means that for any two indexes $t_1, t_2 \in T$ there does not exist an isomorphism $\phi : L_{t_1} \to L_{t_2}$ with $(\phi \circ \pi_{t_1})s = \pi_{t_2} s$ for all $s \in S$. A consequence of this assumption is that T is finite (simply because there are only finitely many distinct mappings from X to D_2 or M_3).

Since we are discussing a subdirect product, the projections $\pi_t : S \to L_t$ $(t \in T)$ have to be onto. This means that the image of the set X under π_t must generate L_t. For $L_t = M_3$ we infer $\pi_t[X] \supseteq \{x, y, z\}$, for $L_t = D_2$ this means $\pi_t[X] = \{0, 1\}$.

From the assumption $g \not\nearrow m$ and from Lemma 1 we obtain:

$$(\gamma g)_* = \mu m \wedge \gamma g.$$

The analogue holds for n and p. By applying the projection π_t we get:

$$((\gamma g)_*)_t = (\mu m)_t \wedge (\gamma g)_t = (\mu n)_t \wedge (\gamma g)_t = (\mu p)_t \wedge (\gamma g)_t. \tag{1}$$

For the case $L_t = M_3$ this equation can only hold if the element $1 \in L_t$ is not contained in $\pi_t[X]$. For the case $L_t = D_2$ this means

$$(\gamma g)_t = 1 \Longrightarrow (\mu m)_t = (\mu n)_t = (\mu p)_t = 0.$$

Table 1 presents all possibilities of assigning elements of M_3 or D_2 to the members of X by any projection π_t such that all the above conditions are met. By

Table 1. The possible candidates for projections π_t given by their image of $\mu m, \mu n, \mu p$ and γg

t	V_t	$(\mu m)_t$	$(\mu n)_t$	$(\mu p)_t$	$(\gamma g)_t$
1		x	x	y	z
2		x	y	x	z
3		y	x	x	z
4	M_3	0	x	y	z
5		x	0	y	z
6		x	y	0	z
7		x	y	z	0
8		1	1	1	0
9		1	1	0	0
10		1	0	1	0
11	D_2	1	0	0	0
12		0	1	1	0
13		0	1	0	0
14		0	0	1	0
15		0	0	0	1

permutation of the elements $x, y, z \in M_3$ one could get further projections – however, they would yield redundant factors and are therefore not listed in the table.

The index set T must now be chosen as a subset of $\{1, \ldots, 15\}$ and in the last part of the proof we show that for none of these choices S is a subdirect product of the lattices L_t. Obviously, the smallest element in S is

$$(0)_{t \in T} = \mu m \wedge \mu n \wedge \mu p \wedge \gamma g.$$

By Equation 1 this yields $(0)_{t \in T} = (\gamma g)_*$. Since γg is an upper neighbor of $(\gamma g)_*$ in S (Lemma 1), there must be at least one index $t \in T$ such that $(\gamma g)_t \neq 0$. Thus, T contains at least one of the indexes 1 through 6 or 15. For each such t we derive a contradiction. Let a and b be two of the attributes in $\{m, n, p\}$. According to Lemma 1, $g \not\swarrow a$ yields $\mu a \neq (\mu a)^* = \mu a \vee \gamma g \in S$. Since $(\mu a)^*$ is the only upper neighbor of μa in L and thus also in S, it follows from $\mu a \vee \mu b \geq \mu a$ that

$$\forall a, b \in \{m, n, p\} : \mu a \vee \mu b = \mu a \quad \text{or} \quad \mu a \vee \mu b \geq \mu a \vee \gamma g. \tag{2}$$

If t is one of the indexes $1, 2, 3$ or 15, we can always choose two distinct attributes a, b from $\{m, n, p\}$ such that $(\mu a)_t = (\mu b)_t$ and yield

$$(\mu a)_t \vee (\gamma g)_t > (\mu a)_t \vee (\mu b)_t < (\mu b)_t \vee (\gamma g)_t. \tag{3}$$

Thus, only the first part of Condition 2 can be true and we obtain $\mu a = \mu a \vee \mu b = \mu b$. Since (G, M, I) is clarified, this contradicts the distinctness of a and b and therefore neither of the four indexes can be in T. If t is one of the remaining indexes $4, 5, 6$, we can always choose $a \neq b$ such that $(\mu a)_t = 0$. In all these cases we then have that $\gamma g \vee \mu a$ and $\mu b \vee \mu a$ are incomparable in S and $\mu a \vee \mu b \neq \mu a$, contradicting Condition 2.

We have thus eliminated all indexes that would allow $(\gamma g)_t \neq 0$ and thus falsified the initial assumption. The second part of the proposition follows dually.

5 Projective Lattices and Covers of Lattices

For the second part of the proof of Theorem 1 we employ the theory of projective lattices. Therefore, we quickly recall some further definitions and results. The more general definitions of theses notions for arbitrary algebras can be found in [2].

Definition 1. *A lattice L in a variety of lattices $\mathbf{K}$ is called* projective *in $\mathbf{K}$, if for any two lattices $L_1, L_2 \in \mathbf{K}$ and any surjective lattice homomorphism $\phi : L_1 \twoheadrightarrow L_2$ and any lattice homomorphism $\psi : L \to L_2$ there exists a lattice homomorphism $\tilde{\psi} : L \to L_1$ such that $\phi \circ \tilde{\psi} = \psi$ holds.*

The following lemma explains a useful property of such projective lattices.

Lemma 4. *If a lattice L_1 is projective in a lattice variety $\mathbf{K}$ and $\phi : L_2 \twoheadrightarrow L_1$ is a surjective homomorphism from a lattice $L_2 \in \mathbf{K}$ onto L_1, then L_2 contains a sublattice S which is isomorphic to L_1 such that its image under ϕ is L_1.*

Proof. Let $\psi : L_1 \to L_1, x \mapsto x$ denote the identity homomorphism on L_1. Since L_1 is projective in $\mathbf{K}$, there is a homomorphism $\tilde{\psi} : L_1 \to L_2$ with $\phi \circ \tilde{\psi} = \psi$. Then $S := \tilde{\psi}[L_1]$ is a sublattice of L_2 and we obtain

$$L_1 = \psi[L_1] = \phi[\tilde{\psi}[L_1]] = \phi[S].$$

Since $\phi \circ \tilde{\psi}$ is the identity map on L_1, the homomorphism $\tilde{\psi}$ must be one-to-one and is thus an isomorphism between L_1 and $S = \tilde{\psi}[L_1]$.

Definition 2. *Let $\mathbf{K}$ be a variety of lattices containing lattices L_1 and L_2 and let $\omega : L_1 \twoheadrightarrow L_2$ be a surjective homomorphism. The pair (L_1, ω) is called a* cover *of L_2 in $\mathbf{K}$, if for any lattice $L \in \mathbf{K}$ and each homomorphism $\tau : L \to L_1$ holds, that if $\omega \circ \tau$ is surjective onto L_1, then τ itself is surjective. Further, (L_1, ω) is called a* projective cover *of L_2, if L_1 is also projective in $\mathbf{K}$.*

The next theorem is a special case of Theorem 5.1 in [2] that provides us with a cover of the lattice $M_{3,3}$.

Theorem 5. *There exists only one surjective homomorphism $\omega : \bar{M}_{3,3} \twoheadrightarrow M_{3,3}$ from $\bar{M}_{3,3}$ onto $M_{3,3}$. In the variety of all modular lattices, $(\bar{M}_{3,3}, \omega)$ is a projective cover of $M_{3,3}$.*

Using the element names of Figure 2, the homomorphism ω is given by

$$\omega : \bar{M}_{3,3} \to M_{3,3} : v \mapsto \begin{cases} x & \text{for } v = \bar{x} \\ y & \text{for } v = \bar{y} \\ v & \text{else.} \end{cases}$$

Corollary 2. *If $\phi : M \twoheadrightarrow M_{3,3}$ is a surjective homomorphism from a modular lattice M onto $M_{3,3}$, then M contains a sublattice S such that the image of S under ϕ is $M_{3,3}$ with S being isomorphic to either $M_{3,3}$ or $\bar{M}_{3,3}$.*

Proof. Since $\phi : M \twoheadrightarrow M_{3,3}$ is surjective and $\bar{M}_{3,3}$ is projective (Theorem 5), by definition there exists a homomorphism $\tilde{\omega} : \bar{M}_{3,3} \to M$ such that $\phi \circ \tilde{\omega} = \omega$ holds. Let S be the image of $\bar{M}_{3,3}$ under $\tilde{\omega}$. Then S is a sublattice of M and we obtain

$$\phi[S] = (\phi \circ \tilde{\omega})[\bar{M}_{3,3}] = \omega[\bar{M}_{3,3}] = M_{3,3}.$$

The lattice $\bar{M}_{3,3}$ has (up to isomorphism) only four distinct homomorphic images: $\bar{M}_{3,3}$, $M_{3,3}$, D_2 and the lattice with only one element. Thus, S is isomorphic to one of those four lattices and since it must be large enough to allow $\phi[S] = M_{3,3}$, it must be isomorphic to either $\bar{M}_{3,3}$ or $M_{3,3}$.

Finally, before we start proving the second part of our theorem we quote Lemma 4 from [5], that makes direct use of the above Lemma 4.

Lemma 5. *Let $\mathfrak{M}^2$ be the variety generated by all modular lattices of length two. Then M_4 is projective in $\mathfrak{M}^2$, i. e. for any surjective homomorphism $\phi : M \twoheadrightarrow M_4$ with $M \in \mathfrak{M}^2$ there is a sublattice S of M that is isomorphic to M_4 and its image under ϕ is M_4.*

6 Characterization Part II

Proposition 3. *If (G, M, I) is the context of a modular, doubly founded complete lattice L and if for each object $g \in G$ holds $|\{m \in M \mid g \not\nearrow m\}| \leq 2$, then L is a member of the variety $\mathbf{M}_3$.*

Proof. The variety $\mathbf{M}_3$ is the smallest modular non-distributive lattice variety and it has only one lower neighbor in the lattice of all modular varieties. Due to Corollary 1 there are three possible situations:

$$\mathrm{Var}(\{L\}) \supseteq \mathbf{M}_{3,3}, \quad \mathrm{Var}(\{L\}) \supseteq \mathbf{M}_4 \quad \text{or} \quad L \in \mathbf{M}_3.$$

In the following we will contradict the first two inclusions.
Assumption 1: $\mathrm{Var}(\{L\}) \supseteq \mathbf{M}_{3,3}$.
From the **HSP**-Theorem follows that $M_{3,3}$ is in $\mathbf{HSP}(\{L\})$. Thus, Corollary 2 yields that $\bar{M}_{3,3}$ is isomorphic to a member of $\mathbf{SP}(\{L\})$ or that $M_{3,3}$ is isomorphic to member of $\mathbf{SP}(\{L\})$. We eliminate both cases.

First, let P be a power of L that contains a sublattice S isomorphic to $\bar{M}_{3,3}$. The elements of S shall be named $u, a, b, c, \bar{c}, \bar{x}, x, y, z, v$ like in Figure 2. Furthermore, let $\mathbb{K}_P$ be the direct context sum that belongs to P. I. e. if $P = \prod_{t \in T} L$ and $(G_t, M_t, I_t) = (G, M, I)$, then $\mathbb{K}_P$ is the context

$$(G_P, M_P, I_P) := \Big(\bigcup_{t \in T} \dot{G}_t, \bigcup_{t \in T} \dot{M}_t, \bigcup_{t \in T} \dot{I}_t \cup \bigcup_{t_1 \in T} (\dot{G}_{t_1} \times \bigcup_{t_2 \in T \setminus \{t_1\}} \dot{M}_{t_2}) \Big).$$

Herein, the dot notion as usual means disjoint union, thus technically $\dot{G}_t$ means $G \times \{t\}$ (for details see [6], Definition 8). Now P is isomorphic to $\underline{\mathfrak{B}}(\mathbb{K}_P)$ (cf. [6], Definition 32) and for the sake of convenience, we assume that $P = \underline{\mathfrak{B}}(\mathbb{K}_P)$. $\mathbb{K}_P$ inherits the arrow-relations from its summands, i. e.

$$(g, t_1) \nearrow (m, t_2) \iff t_1 = t_2 \text{ and } g \nearrow m \text{ in } (G, M, I)$$

and analogously for the $\swarrow$ relation. According to Lemma 3, P is doubly founded, since L is. Therefore, an element $s \in P$ can be chosen such that it is minimal with respect to the properties $s \leq a$ and $s \not\leq u$.

This element s is supremum-irreducible in P because, if we assumed $s = \sup Q$ for some subset $Q \subseteq P$, then there would exist an element $q \in Q$ with $q \not\leq u$ – otherwise we would obtain $s = \sup Q \leq u$ in contradiction to the condition $s \not\leq u$. However, this would mean $q \leq s \leq a$ and from the minimality of s would follow $s = q \in Q$. From that irreducibility follows for s that there must exist an irreducible object g in $\mathbb{K}_P$ with $s = \gamma g$.

For g we can now find three attributes m_b, m_y, m_z in $\mathbb{K}_P$ such that g has double arrows to all three of them. The following construction of the attribute m_z is visualized in Figure 3.

Since g is irreducible, we obtain

$$s = \gamma g > (\gamma g)_* =: s_*$$

and from the minimality condition of s also $s_* \leq u$ and thus $s \wedge u = s_*$. Due to the latter equation, the Isomorphism Theorem (Theorem 2) applies to the intervals $[s_*, s]$ and $[u, s \vee u]$. We infer that $s \vee u$ is an upper neighbor of u, since s is an upper neighbor of s_*. In S (and thus in P), clearly, we have

$$u = a \wedge z \quad \text{and} \quad v = a \vee z$$

and therefore by the Isomorphism Theorem also $[u, a]$ and $[z, v]$ are isomorphic. Because of $s \leq a$ we have $s \vee u \in [u, a]$ which means that

$$s_z := s \vee u \vee z = s \vee z \in [z, v]$$

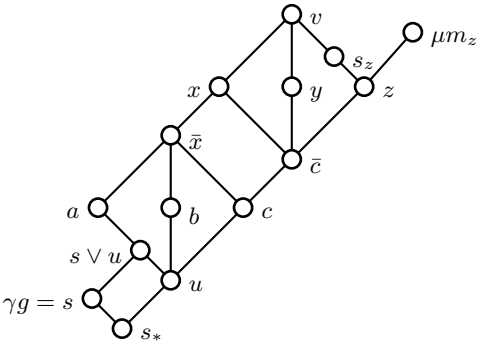


Fig. 3. A sketch to the proof of Proposition 3

is an upper neighbor of z. To find the attribute m_z with $g \swarrow\!\nearrow m_z$, we first consider the set

$$M_z := \{m \in M_P \mid \mu m \geq z, \mu m \not\geq v, m \text{ irreducible}\}.$$

Because of $v > z$ this set cannot be empty. For each attribute $m \in M_z$ further holds

$$s \geq \mu m \wedge s \geq z \wedge s \geq u \wedge s = s_*.$$

Because of the neighborhood relation between s and s_* then either $\mu m \wedge s = s$ or $\mu m \wedge s = s_*$ holds. If we assumed $\mu m \geq s$ for all attributes $m \in M_z$, we would obtain

$$\begin{aligned} z &= \inf\{\mu m \mid m \in M_P, \mu m \geq z\} \\ &= \inf\{\mu m \mid m \in M_P, \mu m \geq v\} \wedge \inf\{\mu m \mid m \in M_P, \mu m \not\geq v, \mu m \geq z\} \\ &= v \wedge \inf \mu[M_z] \\ &\geq v \wedge s = s. \end{aligned}$$

Thus, we would have $z = s \vee z = s_z$ which is in contradiction to the fact that s_z is an upper neighbor of z. Therefore, there must exist at least one object $m_z \in M_z$ with $\mu m_z \wedge s = s_*$, yielding

$$\mu m_z \wedge \gamma g = (\gamma g)_*.$$

From Lemma 1 follows that $g \swarrow m_z$ and since P is modular, $g \swarrow\!\nearrow m_z$ follows by Lemma 2.

Completely analogously one can construct an attribute

$$m_y \in M_y := \{m \in M_P \mid \mu m \geq y, \mu m \not\geq v, m \text{ irreducible}\}$$

with $g \swarrow\!\nearrow m_y$.

Eventually, since Theorem 2 applies also for the intervals $[u, a]$ and $[b, \bar{x}]$, we can find for b an upper neighbor $s_b \in [b, \bar{x}]$ using the same argument as in the case of s_z and $[z, v]$. Again, this gives us an attribute

$$m_b \in M_b := \{m \in M_P \mid \mu m \geq b, \mu m \not\geq \bar{x}, m \text{ irreducible}\}$$

with $g \swarrow\!\nearrow m_b$.

Next, we verify that those three attributes are pairwise distinct. If we assume $m_z = m_y$, we have $m_z \in M_y \cap M_z$ and thus $\mu m_z \geq z \vee y = v$ which implies $m_z \notin M_z$ in contradiction to the construction of m_z. From $m_z = m_b$ we obtain $m_b \in M_b \cap M_z$, thus $\mu m_z \geq \bar{x} \vee z = v$ and again $m_z \notin M_z$. In the same way we can rule out $m_y = m_b$.

In the context $\mathbb{K}_P$ the object g now has three double arrows. By the construction of $\mathbb{K}_P$ as the sum of contexts all three arrows have to be in one summand which means in (G, M, I). This contradicts the requirements of the theorem and therefore the assumption that $\bar{M}_{3,3}$ was isomorphic to a member of $\mathbf{SP}(\{L\})$ cannot be true.

By the same reasoning there is no direct power of L that contains a sublattice isomorphic to $M_{3,3}$ (one only chooses x instead of $\bar{x}$ during the construction of m_b). Thus, also $M_{3,3}$ cannot be a member of $\mathbf{HSP}(\{L\})$ and in summary we obtain $\mathrm{Var}(\{L\}) \not\supseteq \mathbf{M}_{3,3}$.

Assumption 2: $\mathrm{Var}(\{L\}) \supseteq \mathbf{M}_4$. Since we already know that $M_{3,3}$ is not a member of $\mathrm{Var}(\{L\})$, we can apply Theorem 3 and obtain that each lattice in the variety $\mathrm{Var}(\{L\})$ is isomorphic to a subdirect product of lattices of length two or less. By definition this means that $\mathrm{Var}(\{L\}) \subseteq \mathfrak{M}^2$. By the **HSP**-Theorem from Assumption 2 we obtain $M_4 \in \mathbf{HSP}(\{L\})$ and from Lemma 5 we obtain eventually that M_4 is isomorphic to a lattice in $\mathbf{SP}(\{L\})$. Therefore, there is a direct power P of L, that contains a sublattice S isomorphic to M_4. Let the elements of S be named like in Figure 2. Analogously to the procedure above, let $\mathbb{K}_P$ be the context sum with the summands (G, M, I) such that P is isomorphic to the concept lattice of $\mathbb{K}_P$. Due to being doubly founded, $\mathbb{K}_P$ then contains an irreducible object g such that $s := \gamma g$ is an element of L which is minimal with respect to the properties $s \leq a$ and $s \not\leq u$. Again, we find three distinct attributes

$$m_b \in M_b := \{m \in M_P \mid \mu m \geq b, \mu m \not\geq v, m \text{ irreducible}\},$$

and $m_c \in M_c$, $m_d \in M_d$ defined analogously satisfying $g \nearrow m_b$, $g \nearrow m_c$ and $g \nearrow m_d$.

Since $\mathbb{K}_P$ is a context sum, all three attributes belong to the same summand and thus g has three double arrows in (G, M, I). Since this is a contradiction to the proposition's requirements, also Assumption 2 must be false.

The only possible alternative left now is $\mathrm{Var}(\{L\}) \subseteq \mathbf{M}_3$ and so L is a member of the variety $\mathbf{M}_3$.

By duality we get the same result if we require in Proposition 3 that $|\{g \in G \mid g \nearrow m\}| \leq 2$ holds for any attribute $m \in M$ instead of $|\{m \in M \mid g \nearrow m\}| \leq 2$ for any object $g \in G$. This concludes our proof of the claimed characterization in Theorem 1.

Remark: Finally, it is worth noting that an easy adaption of the characterization to other lattice varieties – by allowing more arrows per object or per attribute to cover larger varieties – is not possible, as the following example for the variety

$M_{3,3}$	a	b	c	d	e
1	↗	↗	×	↗	×
2	↗	↗	×	×	↗
3	×	×	×	↗	↗
4	↗	×	↗		
5	×	↗	↗		

Fig. 4. A context of $M_{3,3}$, containing three or less arrows in each row and column.

$\mathbf{M}_4$ demonstrates: It is well known that the (modular and doubly founded) lattice $M_{3,3}$ is not a member of the variety $\mathbf{M}_4$. However, the context given in Figure 4 is clarified, has only three arrows per row and column and its concept lattice is isomorphic to $M_{3,3}$.

References

1. Birkhoff, G.: Lattice theory, 3rd edn. Colloquium Publications, vol. 25. American Mathematical Society, Providence (1967)
2. Day, A.: Characterizations of finite lattices that are bounded-homomorphic images of sublattices of free lattices. Canadian Journal of Mathematics 31, 69–78 (1979), http://cms.math.ca/cjm/v31/p69#
3. Erné, M.: Distributive laws for concept lattices. Algebra Universalis 30, 538–580 (1993), http://dx.doi.org/10.1007/BF01195382
4. Faigle, U., Herrmann, C.: Projective geometry on partially ordered sets. Transactions of the American Mathematical Society 266(1), 319–332 (1981), http://www.jstor.org/stable/1998401
5. Ganter, B., Poguntke, W., Wille, R.: Finite sublattices of four-generated modular lattices. Algebra Universalis 12(1), 160–171 (1981), http://dx.doi.org/10.1007/BF02483876
6. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
7. Garcia, O., Nelson, E.: On the nonexistence of free complete distributive lattices. Order 1(4), 399–403 (1985), http://dx.doi.org/10.1007/BF00582745
8. Grätzer, G.: General Lattice Theory, 2nd edn. Birkhäuser, Basel (1998)
9. Hales, A.W.: On the non-existence of free complete boolean algebras. Fundamentae Mathematica 54, 45–66 (1964), http://matwbn.icm.edu.pl/tresc.php?wyd=1&tom=54
10. Jónsson, B.: Algebras whose congruence lattices are distributive. Mathematica Scandinavica 21, 110–121 (1967)
11. Jónsson, B.: Equational classes of lattices. Mathematica Scandinavica 22, 187–196 (1968)

Factorization with Hierarchical Classes Analysis and with Formal Concept Analysis

Cynthia Vera Glodeanu

Technische Universität Dresden,
01062 Dresden, Germany
Cynthia_Vera.Glodeanu@mailbox.tu-dresden.de

Abstract. We present a comparison between Hierarchical Classes Analysis and the formal concept analytical approach to Factor Analysis regarding the factorization problem of binary matrices. Both methods decompose a binary matrix into the Boolean matrix product of two binary matrices such that the number of factors is as small as possible. We show that the two approaches yield the same decomposition even though the methods are different. The main aim of this paper is to connect the two fields as they produce the same results and we show how the two domains can benefit from one another.

Keywords: Hierarchical Classes Analysis, Formal Concept Analysis, Factor Analysis.

1 Introduction

We compare two methods regarding the decomposition of a binary matrix. The formal concept analytical approach to Factor Analysis was developed in [1]. The factors resulting from this approach are formal concepts, and they were shown to yield an optimal factorization. The Hierarchical Classes Analysis uses object and attribute bundles in the decomposition, which are in fact extents and intents of concepts.

Formal Concept Analysis and Hierarchical Classes Analysis were first linked in [2] however it did not include the factorization problem.

Section 2 and 3 contain brief introductions to Formal Concept Analysis and to the formal concept analytical approach to Factor Analysis, respectively. In Section 4 we present the basic notions of Hierarchical Class Analysis, their translation into the language of Formal Concept Analysis and the main differences between the two fields. We also give a small example for this purpose. Section 5 contains the main results showing that both methods yield the same results but they differ through the mathematical and algorithmic approaches. We also comment on the generalization of the two fields regarding real, fuzzy and triadic data. Section 6 contains concluding remarks and future work.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 107–118, 2011.

2 Formal Concept Analysis

The mathematical foundation of Formal Concept Analysis was developed by the research group *Allgemeine Algebra und Diskrete Mathematik* around Rudolf Wille at the Technical University Darmstadt as an instrument for data analysis at the beginning of the 80s.

We give a brief introduction to Formal Concept Analysis, containing definitions and results from [3], and refer the interested reader to this work.

Definition 1. *A **formal context** $\mathbb{K} = (G, M, I)$ consists of two sets G and M and a binary relation $I \subseteq G \times M$. The elements of G are called **objects**, the ones of M **attributes** and $(g, m) \in I$ means that the object g has the attribute m. The relation I is called the **incidence relation** of the context.*

Finite small contexts can be represented through cross tables. The rows of the table are named after the objects and the columns after the attributes. The row corresponding to the object g and the column corresponding to the attribute m contains a cross if and only if $(g, m) \in I$.

To obtain the concepts of the context one has to consider for each object the accurate attributes and for each attribute the objects to which it belongs to.

Definition 2. *For $A \subseteq G$ and $B \subseteq M$ the **derivation operators** are defined as:*

$$A^{\prime} := \{m \in M \mid (g, m) \in I \text{ for all } g \in A\} \text{ and}$$
$$B^{\prime} := \{g \in G \mid (g, m) \in I \text{ for all } m \in B\}.$$

*A **formal concept** of (G, M, I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$ such that $A^{\prime} = B$ and $B^{\prime} = A$. A is called the **extent** and B the **intent** of the concept (A, B). The set of all formal concepts is denoted by $\mathfrak{B}(G, M, I)$.*

Formal concepts correspond to maximal rectangles full of crosses in the cross table representation of a formal context.

For brevity, we write $g^{\prime}$ and $m^{\prime}$ instead of $\{g\}^{\prime}$ and $\{m\}^{\prime}$, respectively.

Definition 3. *The sets*

$$\mathcal{O}(G, M, I) := \{(g^{\prime\prime}, g^{\prime}) \mid g \in G\},$$
$$\mathcal{A}(G, M, I) := \{(m^{\prime}, m^{\prime\prime}) \mid m \in M\}$$

*are called **object concepts** and **attribute concepts**, respectively.*

The derivation operators form a Galois connection between the (power sets of the) sets G and M and their compounds form closure operators.

Concepts serve for classification. Consequently the super- and subrelation play an important role. A concept is called superconcept of another if it is more general, if it contains more objects.

Definition 4. *Let (A, B) and (C, D) be two concepts of a context (G, M, I). (C, D) is called* ***superconcept*** *of (A, B), $(A, B) \leq (C, D)$, iff $A \subseteq C$ $(D \subseteq B)$. Then (A, B) is called* ***subconcept*** *of (C, D). The relation $\leq$ is called* ***hierarchical order*** *(or simply* ***order****) of the concepts. The set of all concepts of (G, M, I) ordered in this way is denoted by $\underline{\mathfrak{B}}(G, M, I)$ and is called* ***concept lattice*** *of the context (G, M, I).*

Theorem 1. ***Basic Theorem of Formal Concept Analysis***
The concept lattice of any formal context (G, M, I) is a complete lattice. The supremum and infimum are given by:

$$\bigwedge_{t\in T}(A_t, B_t) = \left(\bigcap_{t\in T} A_t, \left(\bigcup_{t\in T} B_t\right)''\right),$$

$$\bigvee_{t\in T}(A_t, B_t) = \left(\left(\bigcup_{t\in T} A_t\right)'', \bigcap_{t\in T} B_t\right).$$

In general, a complete lattice L is isomorphic to $\underline{\mathfrak{B}}(G, M, I)$ iff there exist mappings $\tilde{\gamma} : G \to L$ and $\tilde{\mu} : M \to L$ such that $\tilde{\gamma}(G)$ is $\bigvee$-dense in L, $\tilde{\mu}(M)$ is $\bigwedge$-dense in L and $gIm \Leftrightarrow \tilde{\gamma}g \leq \tilde{\mu}m$ for $g \in G$ and $m \in M$. In particular, $L \cong \underline{\mathfrak{B}}(L, L, \leq)$.

3 Factor Analysis through Formal Concept Analysis

This section contains the main results from [1] and the translation of the problem to Formal Concept Analysis. In [1] an approach to Factor Analysis is presented: A $p \times q$ binary matrix W is decomposed into the Boolean matrix product $P \circ Q$ of a $p \times n$ binary matrix P and an $n \times q$ binary matrix Q with n as small as possible. The Boolean matrix product $P \circ Q$ is defined as $(P \circ Q)_{ij} = \bigvee_{l=1}^{n} P_{il} \cdot Q_{lj}$, where $\bigvee$ is the maximum and $\cdot$ is the product. Through the Boolean matrix product a non-linear relationship between objects, factors and attributes is given.

The matrices W, P and Q represent an object-attribute, object-factor and factor-attribute relationship, respectively. $W = P \circ Q$ means that object i is incident with attribute j if and only if there is a factor l which applies to i and contains j.

The method developed in [1] uses formal concepts as factors which produce decompositions with smallest number of factors possible. Contexts can be seen as Boolean matrices by replacing in the cross table the crosses by 1's and the blanks by 0's.

In the language of Formal Concept Analysis we give the following definition for factors:

Definition 5. *A subset of formal concepts $\mathcal{F} \subseteq \mathfrak{B}(G, M, I)$ such that*

$$I = \bigcup_{(A,B)\in\mathcal{F}} A \times B$$

is called ***factorization****. If* $\mathcal{F}$ *is minimal with respect to its cardinality then it is called* ***optimal factorization****. The elements of* $\mathcal{F}$ *are called* ***(optimal) factors****.*

Note that an optimal factorization is not always unique, since different subsets of formal concepts with the same cardinality may cover the incidence relation in a formal context.

For a subset $\mathcal{F} = \{(A_1, B_1), \ldots, (A_n, B_n)\} \subseteq \mathfrak{B}(G, M, I)$ of formal concepts we construct binary matrices $A_\mathcal{F}$ and $B_\mathcal{F}$ as follows:

$$(A_\mathcal{F})_{il} = \begin{cases} 1 \text{ if } i \in A_l \\ 0 \text{ if } i \notin A_l \end{cases}, \qquad (B_\mathcal{F})_{lj} = \begin{cases} 1 \text{ if } j \in B_l \\ 0 \text{ if } j \notin B_l \end{cases},$$

where $i \in \{1, \ldots, |G|\}, j \in \{1, \ldots, |M|\}, l \in \{1, \ldots, n\}$ and A_l and B_l are the extent and intent of the l-th factor, respectively.

We consider a decomposition of the Boolean matrix W associated to (G, M, I) into the Boolean matrix product $A_\mathcal{F} \circ B_\mathcal{F}$.

Theorem 2 ***(Universality of concepts as factors [1]).*** *For every* W *there is* $\mathcal{F} \subseteq \mathfrak{B}(G, M, I)$ *such that* $W = A_\mathcal{F} \circ B_\mathcal{F}$.

Theorem 3 ***(Optimality of concepts as factors [1]).*** *Let* $W = P \circ Q$ *for* $p \times n$ *and* $n \times q$ *binary matrices* P *and* Q*. Then there exists a subset* $\mathcal{F} \subseteq \mathfrak{B}(G, M, I)$ *of formal concepts of* W *with* $|\mathcal{F}| \leq n$ *such that for the* $p \times |\mathcal{F}|$ *and* $|\mathcal{F}| \times q$ *binary matrices* $A_\mathcal{F}$ *and* $B_\mathcal{F}$ *we have* $W = A_\mathcal{F} \circ B_\mathcal{F}$.

The proofs are based on the fact that a binary matrix can be seen as a $\bigvee$-superposition of rectangles full of crosses. Each such rectangle is contained in a maximal rectangle. Every maximal rectangle full of crosses corresponds to a formal concept.

Theorem 4 ***(Mandatory factors [1]).*** *If* $W = A_\mathcal{F} \circ B_\mathcal{F}$ *for some subset* $\mathcal{F} \subseteq \mathfrak{B}(G, M, I)$ *of formal concepts then* $\mathcal{O}(G, M, I) \cap \mathcal{A}(G, M, I) \subseteq \mathcal{F}$*, where* $\mathcal{O}(G, M, I)$ *and* $\mathcal{A}(G, M, I)$ *are the sets of object and attribute concepts, respectively.*

The proof is based on the fact that if one considers a formal concept $(A, B) \in \mathcal{O}(G, M, I) \cap \mathcal{A}(G, M, I)$, then $(A, B) = (g'', g') = (m', m'')$ for some $g \in G$ and $m \in M$. The formal concept (A, B) is the only one which contains the tuple (g, m).

Theorem 5 ***[1,4].*** *The problem to find a decomposition* $W = P \circ Q$ *of an* $p \times q$ *binary matrix* W *into a* $p \times n$ *binary matrix* P *and an* $n \times q$ *binary matrix* Q *with* n *as small as possible is NP-hard and the corresponding decision problem is NP-complete.*

However there is a greedy approximation algorithm presented in [1] which computes the optimal factorization of a Boolean matrix by choosing the concepts

which cover most of the incidence data. Due to its greedy approach the algorithm is applicable on large data sets but the disadvantage lies in the fact that it may provide suboptimal solutions. In [1] the authors generated 80,000 binary matrices of size 20×20 with 50% density of 1's and compared the results obtained by the greedy algorithm with the results obtained by a brute force algorithm. The test yields that the average number of factors determined by the greedy algorithm is close to the optimal number of factors. For example, if the optimal number of factors is 5 the greedy algorithm yields 5.255 ± 0.473 (average value $\pm$ standard deviation) factors and in the case of 11 optimal factors it produces 12.740 ± 1.290 factors. For the whole statistic we refer to [1].

3.1 Approximate Factorization

In case of large contexts the number of optimal factors can be quite big. The approximate factorization requires that the factors from $\mathcal{F} \subseteq \mathfrak{B}(G, M, I)$ cover just a part of the incidence relation. By applying approximate factorization to our data we look for matrices P and Q such that W, the corresponding binary matrix to the context (G, M, I), is approximately equal to $P \circ Q$. By adding further concepts to the approximate factorization $\mathcal{F}$, we obtain a more precise approximation of the data. While exact factorization may require a large number of factors, a considerably smaller number of factors may cover a large portion of the data.

This kind of approximate factorization yields *negative discrepancies*, i.e. since $P \circ Q$ is approximately equal to W there exist entries where $(P \circ Q)_{ij} = 0$ and $W_{ij} = 1$.

For obtaining an approximate factorization with *positive discrepancies*, i.e. $(P \circ Q)_{ij} = 1$ and $W_{ij} = 0$, we have to compute dense rectangles instead of full rectangles, which correspond to formal concepts. The computation of dense rectangles of a given formal context was studied in [5,6].

4 Hierarchical Classes Analysis

The theory of Hierarchical Classes Analysis was developed by De Boeck and Rosenberg at the end of the 80's. In this section we give a brief introduction to this field and translate the notions of Hierarchical Classes Analysis into the language of Formal Concept Analysis. The interested reader may find a more detailed introduction to Hierarchical Classes Analysis in [7,8].

Since the development of Hierarchical Classes Analysis there is recent work going on for the generalization of the model for non-binary data and non-dyadic data. A detailed discussion is done in the next section.

In the following we just give the definitions and notions for objects, the ones for attributes can be done analogously by interchanging objects with attributes and vice versa.

The *object by attribute data* from Hierarchical Classes Analysis is a binary matrix which corresponds to the Boolean matrix representation of a formal context. In Hierarchical Classes Analysis two objects are called *equivalent* if and

Table 1.

	a	b	c	d	e	f	g	h
1	×	×	×	×	×	×	×	×
2	×	×	×	×	×	×	×	×
3	×	×	×	×			×	×
4	×	×	×	×			×	×
5			×	×	×	×	×	×
6	×	×					×	×
7	×	×					×	×
8					×	×	×	×
9								
10								

only if they have the same attributes. An *object class* is the set of all objects that are equivalent to one another in a given object by attribute data. An object class is denoted by square brackets, i.e. if $g_1, \ldots, g_n$ form an object class, we write $[g_1, \ldots, g_n]$. The class of objects to which none attributes, the empty set of attributes, apply is called the *undefined class*.

In the context given in Table 1 the objects 1 and 2 are equivalent and therefore form an object class. There are six object classes, namely $[1, 2], [3, 4], [5], [6, 7]$, $[8]$ and $[9, 10]$ where the last one is an undefined class, because no attribute applies to its objects. The attribute classes are $[a, b], [c, d], [e, f]$ and $[g, h]$. In this example we do not have any undefined attribute class.

Each object class is characterised by the set of attributes that apply to all objects of the class. Therefore the object classes can be ordered by the super-/subset relation of their attribute sets. This order is a partial one, called the *hierarchy of object classes*.

In the language of Formal Concept Analysis we give the following definitions of the notions presented above:

Definition 6. *In a formal context (G, M, I) two objects $g_1, g_2 \in G$ are called* ***equivalent*** *iff $g_1^{\shortmid} = g_2^{\shortmid}$. The set*

$$[g_1] := \{g \in G \mid g^{\shortmid} = g_1^{\shortmid}\}$$

is called ***object class*** *of the object $g_1 \in G$. For the object classes corresponding to the objects $g_1, g_2 \in G$ we define*

$$[g_1] \leq [g_2] :\Longleftrightarrow g_1^{\shortmid} \subseteq g_2^{\shortmid}$$

where $\leq$ is a partial order relation between the classes of G, the so called ***hierarchy of object classes****.*

A simultaneous graphical representation of the hierarchy of object and of attribute classes can be done by using the *association relation* between the two classes. An object class is associated to all attribute classes that apply to the objects of that object class. By associating an object class with an attribute class,

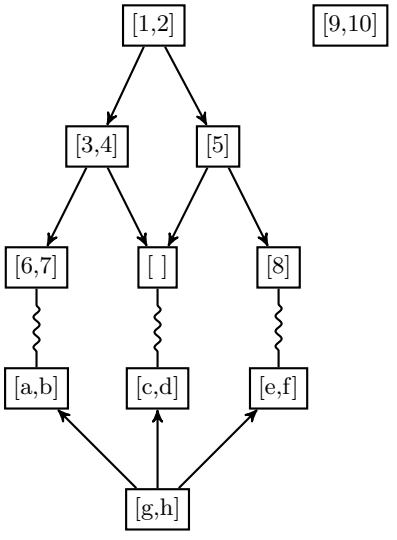


Fig. 1. Hierarchical class representation

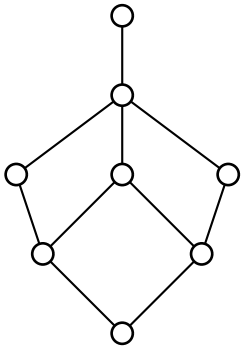


Fig. 2. Concept lattice

the first is also associated to all the superordinated classes of the second one and vice versa. Obviously the association relation is symmetrical. Therefore it is enough to associate the bottom classes of the two hierarchies. Graphically, one hierarchy is represented upside down and the association relation by zigzag lines. The graphical representation of the context from Table 1 is given in Figure 1 and the concept lattice in Figure 2.

The object (attribute) classes are ordered through the super-/subset relation of their attribute (object) sets. However the undefined object (attribute) class is not included into the hierarchy of objects (attributes), because it interferes with the graphical representation. Including the undefined classes into the hierarchies would make them the bottom classes and through the association relation of the bottom classes, every object class would be associated to every attribute class.

In the graphical representation sometimes empty classes are needed for the correct representation of the association relation. For example, in the hierarchy of object classes we have an empty bottom class, because the classes $[3, 4]$ and $[5]$ both apply to the attribute class $[c, d]$. A direct zigzag line from $[3, 4]$ and $[5]$ to $[c, d]$ is not permitted because the two object classes are not bottom classes.

The hierarchical class representation contains the object concepts and the attribute concepts, sometimes however it also contains different concepts if they are needed for the correct representation of the association relation or/and for the optimal factorization. Whereas the concept lattice contains all the concepts of a given context.

The object set which corresponds to an attribute class can be decomposed into object classes such that their size is maximal and their number minimal. The objects contained in such a maximal set are called *object bundles*. An object bundle is in fact a set of objects which is associated to the same bottom class

of attributes. The three relations of the model can be reconstructed from the bundles. A *bundle specific class* is a class that belongs to only one bundle, thus they are the bottom classes in the hierarchy of object (attribute) classes.

For the context given in Table 1 the object bundles are $\{1,2,3,4,6,7\}$, $\{1,2,3,4,5\}$ and $\{1,2,5,8\}$ and the attribute bundles are $\{a,b,g,h\}$,$\{c,d,g,h\}$ and $\{e,f,g,h\}$.

In the language of Formal Concept Analysis we give the following definitions of bundles:

Definition 7. *An **object (attribute) bundle** is the extent (intent) of some concept.*

In Hierarchical Classes Analysis a $p \times q$ binary matrix W is decomposed into the Boolean matrix product $X \circ Y^{\mathcal{T}}$ of a $p \times k$ binary matrix X and a $q \times k$ binary matrix Y with k being the Schein rank of W, i.e. the smallest possible value such that $W = X \circ Y^{\mathcal{T}}$, and $Y^{\mathcal{T}}$ the transpose matrix of Y. The matrices X and Y have as columns the characteristic vectors of the object and attribute bundles, respectively.

As described in [7,9] the Hiclas algorithm computes for a binary matrix W the best fitting Hierarchical Classes model for a given solution rank k. The algorithm assumes that $W = Z + E$, where W, Z and E are $p \times q$ matrices, $Z = X \circ Y^{\mathcal{T}}$ with X and Y being $p \times k$ and $q \times k$ binary matrices, respectively, and E is the discrepancies matrix. X and Y are estimated by a least square approach.

The user must specify the number of bundles, the rank, of the solution. The algorithm starts an iterative procedure based on either an initial set of attribute or object bundles. The initial bundle set can be determined by: 1) a rational heuristic; 2) a random generation procedure; or 3) user provided; where the first two are built into the algorithm. Hiclas can also be used in a confirmatory way through method 3). The algorithm stops either when the pre-entered rank is reached or when the number of discrepancies does not decrease in any further iteration. The *optimal number of bundles* is considered to be the number beyond which the discrepancies decrease slightly.

The solution of the Hiclas algorithm depends strongly on the starting point [7].

5 Comparison of the Two Models

The formal concept analytical approach to Factor Analysis as well as the Hierarchical Classes Analysis use sets of formal concepts in the decomposition of a binary matrix but the mathematical formalisation is different. The algorithms of both methods search for the smallest possible subset of formal concepts which covers the incidence relation in a context but their approaches are different. The algorithm presented in [1] is a greedy approximation approach which is efficient but can possibly yield suboptimal solutions, as discussed in Section 3. A greedy approach was considered because the factorization problem in NP hard and this

algorithm can also be applied on large data sets. On the other hand the Hiclas algorithm is based on a branch-and-bound approach and always yields an optimal factorization. A branch-and-bound approach can be applied just on smaller data sets due to its high complexity. Therefore Hiclas was implemented to compute factorizations with up to 15 bundles.

If we modify the greedy algorithm to a brute force one, then both algorithms yield the same number of factors for a given binary matrix.

In [10] the usability of Formal Concept Analysis was not considered as applicable in data analysis in the sense of Hierarchical Classes Analysis due to the lack of decomposition of the context and the complexity of the concept lattice especially when dealing with large data sets and noisy data. By the method presented in [1] a decomposition of the context is possible, which yields the same results as the decomposition using bundles. The algorithm computing the factorization does not imply the computation of all the formal concepts which enables the problem regarding the complexity of the concept lattice.

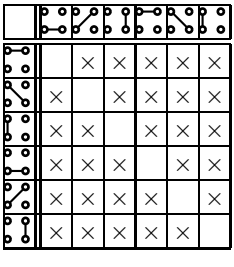
The use of formal concepts in the decomposition provides more possibilities of representing the information contained in the data in the same compact way. This is possible due to the fact that an optimal factorization is sometimes not unique. In [10] the decomposition using bundles was presented as unique up to permutations of rows in the binary matrices X and Y. This is however not the case since both methods use formal concepts and such decomposition is not always unique as discussed in Section 3. Such a non-unique decomposition is presented in Example 1.

For the graphical representation of the hierarchical classes one has to know afore the bundle decomposition whereas the concept lattice can be drawn independently to the factorization.

Example 1. Consider the context below with objects and attributes given by the edges and diagonals of a square. An object is incident with an attribute if and only if their edges have at least a common node.

The context has 64 concepts and at the first glimpse it is difficult to determine how many optimal factors it has. But the context has a complementary set representation and making use of the 1-stepped Ferrers Relations (see [3] for Ferrers Relation) one can easily determine the number of optimal factors. The concept lattice of $(G, M, G \times M \setminus I)$ (Figure 3 left) can be order embedded in the

Table 2. Formal context



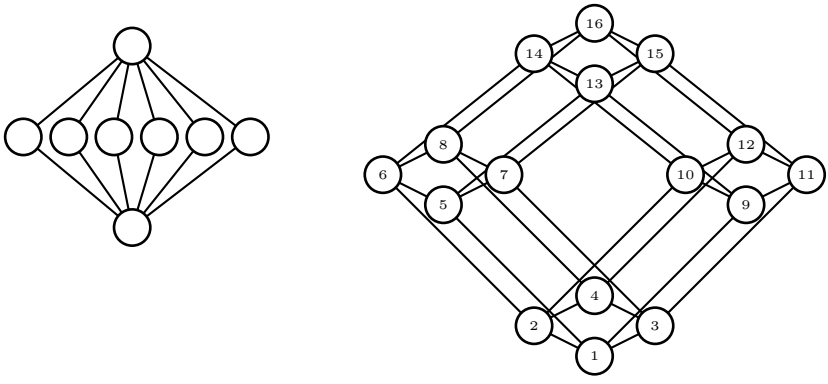


Fig. 3. Order embedding in $(\mathfrak{P}(\{1,2,3,4\}),\leq)$

power set lattice of $(\mathfrak{P}(\{1,2,3,4\}),\leq)$ (Figure 3 right), by identifying the nodes of $\underline{\mathfrak{B}}(G,M,G\times M\setminus I)$ with the nodes $4,6,7,10,11,13$ of the powerset lattice.

This means that the number of optimal factors is 4. By looking at the topmost node at left in the square and all the attributes and objects which contain this node, one gets the first factor of the factorization. By numbering consecutively the attributes and the objects, the first optimal factor is $(\{1,2,3\},\{4,5,6\})$. Proceeding in this manner the other optimal factors are $(\{3,4,5\},\{1,2,6\})$, $(\{1,5,6\},\{2,3,4\})$ and $(\{2,4,6\},\{1,3,5\})$.

This factorization is obtained by using the greedy approximation algorithm proposed in [1].

In the graphical representation using Hierarchical Classes one would tend to draw the object and attribute classes and to associate them correspondingly since each class is a bottom element. This however does not yield an optimal factorization. By knowing the number of bundles needed one could look for the appropriate bundles. The Hiclas algorithm finds also 4 bundles, namely $(\{1,2,3\},\{4,5,6\}),(\{1,5,6\},\{2,3,4\}),(\{2,4,5\},\{1,3,6\}),(\{1,3,6\}),(\{3,4,6\},\{1,2,5\})$. The graphical representation is in Figure 4.

Both methods, formal concept analytical and Hierarchical Classes Analysis approach, yield an optimal factorization but the factors are different. This example is also suitable for the non-uniqueness of the decomposition.

The Hierarchical Class Analysis was extended to data containing positive integers [11] and real values [12]. On the other hand the fuzzy approach to the factorization problem using formal concepts as optimal factors was developed in [13]. How these two models interact with each other and if the results from this paper can be generalised to the approaches presented in [11,12] is topic of a forthcoming paper.

In [14,15] the triadic model of Hierarchical Classes Analysis was presented. The triadic approach using formal concepts was also developed in [16,17,18] and the triadic fuzzy approach in [19]. The comparison of these two methods is also subject of a forthcoming paper.

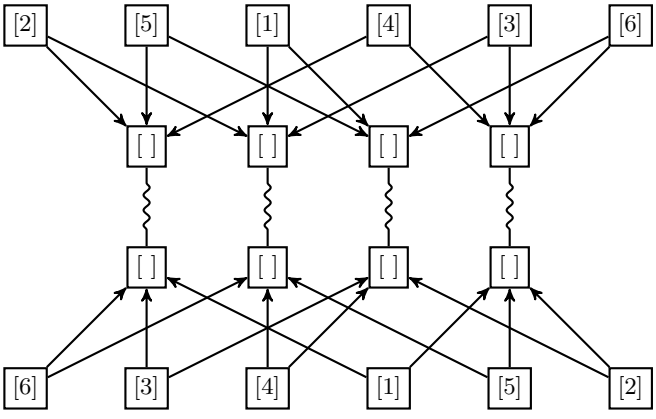


Fig. 4. Hierarchical class representation

6 Conclusion

We compared two methods factorizing binary data. Hierarchical Classes Analysis uses bundles in the decomposition and the formal concept analytical approach to Factor Analysis uses formal concepts. We showed that both methods yield the same factorization, even though the mathematical approaches are different. The Hiclas algorithm performs in general better than the greedy approximation algorithm. However the first one is applicable only on data sets having at most 15 bundles.

The connection between Hierarchical Classes Analysis and Formal Concept Analysis was also studied.

Further research includes the study between the two methods for real and triadic data. We expect that also in these generalisations the domains can benefit from one another.

The results of this paper could convey the interest in Formal Concept Analysis among the Hierarchical Classes Analysis community and vice versa.

References

1. Belohlávek, R., Vychodil, V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. Journal of Computer and System Sciences 76(1), 3–20 (2010)
2. Chen, Y., Yao, Y.: Formal concept analysis and hierarchical classes analysis. In: Fuzzy Information Processing Society, pp. 276–281 (2005)
3. Ganter, B., Wille, R.: Formale Begriffsanalyse: Mathematische Grundlagen. Springer, Heidelberg (1996)
4. Markowsky, G., Nau, D., Woodbury, M.A., Amos, D.: A mathematical analysis of human leukocyte antigen serology. Mathematical Biosciences 40(3-4), 243–270 (1978)
5. Guenoche, A., Mechelen, I.V.: Galois approach to the induction of concepts, ch. 12. Cognitive Science Series, pp. 287–308. Academic Press, London (1993)

6. Belohlávek, R., Vychodil, V.: Dense rectangles in object-attribute data. In: Lin, T.Y., Zhang, Y.Q. (eds.) IEEE International Conference on Granular Computing, pp. 586–591 (2006)
7. De Boeck, P., Rosenberg, S.: Hierarchical classes: model and data analysis. Psychometrika 53, 361–381 (1988)
8. De Boeck, P., Rosenberg, S., Mechelen, I.V.: The Hierarchical Classes Approach: a Review, ch. 11. Cognitive Science Series, pp. 265–286. Academic Press, London (1993)
9. Leenen, I., Mechelen, I.V.: An evaluation of two algorithms for hierarchical classes analysis. Journal of Classification 18, 57–80 (2001)
10. Mechelen, I.V., De Boeck, P.: The conjunctive model of hierarchical classes. Psychometrika 60(4), 505–521 (1995)
11. Mechelen, I.V., Lombardi, I., Ceulemans, E.: Hierarchical classes modeling of rating data. Psychometrika 72, 475–488 (2007)
12. Schepers, J., Mechelen, I.V.: Uniqueness of real-valued hierarchical classes models. Journal of Mathematical Psychology 54(2), 215–221 (2010)
13. Belohlávek, R., Vychodil, V.: Factor analysis of incidence data via novel decomposition of matrices. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 83–97. Springer, Heidelberg (2009)
14. Leenen, I., Mechelen, I.V., De Boeck, P., Rosenberg, S.: Indclas: A three-way hierarchical classes model. Psychometrika 64, 9–24 (1999)
15. Ceulemans, E., Mechelen, I.V., Leenen, I.: Tucker3 hierarchical classes analysis. Psychometrika 68, 413–433 (2003)
16. Krolak-Schwerdt, S., Orlik, P., Ganter, B.: TRIPAT: a model for analyzing three-mode binary data. In: Bock, H.H., Lenski, W., Richter, M.M. (eds.) Studies in Classification, Data Analysis, and Knowledge Organization. Information Systems and Data Analysis, vol. 4, pp. 298–307. Springer, Heidelberg (1994)
17. Belohlávek, R., Vychodil, V.: Optimal factorization of three-way binary data. In: Hu, X., Lin, T.Y., Raghavan, V., Grzymala-Busse, J., Liu, Q., Broder, A. (eds.) GrC, pp. 61–66 (2010)
18. Glodeanu, C.: Triadic factor analysis. In: Kryszkiewicz, M., Obiedkov, S. (eds.) Concept Lattices and Their Applications 2010, pp. 127–138 (2010)
19. Belohlávek, R., Osicka, P.: Triadic concept analysis of data with fuzzy attributes. In: International Conference on Granular Computing, pp. 661–665 (2010)

Gene Expression Array Exploration Using $\mathcal{K}$-Formal Concept Analysis

José María González-Calabozo, Carmen Peláez-Moreno, and Francisco José Valverde-Albacete*

Dpto. de Teoría de la Señal y de las Comunicaciones,
Universidad Carlos III de Madrid
Avda. de la Universidad, 30, Leganés 28911, Spain
{fva,carmen,jmgc}@tsc.uc3m.es

Abstract. DNA micro-arrays are a mechanism for eliciting gene expression values, the concentration of the transcription products of a set of genes, under different chemical conditions. The phenomena of interest—up-regulation, down-regulation and co-regulation—are hypothesized to stem from the functional relationships among transcription products. In [1,2,3] a generalisation of Formal Concept Analysis was developed with data mining applications in mind, $\mathcal{K}$-Formal Concept Analysis, where incidences take values in certain kinds of semirings, instead of the usual Boolean carrier set. In this paper, we use $(\overline{\mathbb{R}}_{\min,+})$- and $(\overline{\mathbb{R}}_{\max,+})$-Formal Concept Analysis to analyse gene expression data for *Arabidopsis thaliana*. We introduce the mechanism to render the data in the appropriate algebra and profit by the wealth of different Galois Connections available in Generalized Formal Concept Analysis to carry different analysis for up- and down-regulated genes.

1 Introduction

The *transcriptome* of a species is the set of gene expression products, be they proteins or messenger RNA (mRNA) chains. DNA micro-arrays are a mechanism to take measures of such data in the form of an *expression profile*, a record of the concentration of different mRNA associated to a subset of the species genome with respect to a *condition*, a particular state or sequence of states undergone by the cells under study. Roughly, each of these mRNA sequences comes from the expression of a particular gene and is translated into a protein inside ribosomes.

Transcriptomics studies these expression profiles for multiple purposes: *body maps*—creating records of baseline abundance of mRNA in different tissues—, *case vs. control studies*—studying particular states vs a control profile—, *parsing pathways*—elucidating the signalling networks associated to sets of genes— and

* This work has been partially supported by the Spanish Government-Comisión Interministerial de Ciencia y Tecnología projects 2008-06382/TEC and 2008-02473/TEC and the regional projects S-505/TIC/0223 (DGUI-CM) and CCG08-UC3M/TIC-4457 (Comunidad Autónoma de Madrid - UC3M).

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 119–134, 2011.

studying *functional response patterns*, the exploration of a systematically varied set of conditions in the expectation that *co-regulation* of genes across a set of biological conditions reveals functional gene groups [4].

In this context, the concentration of the transcribed product (usually mRNA) is the *(gene) expression value*, and the expression values of a set of genes under the same condition an *expression profile*. Therefore, given a *genome* —a set of *genes*—$G = \{g_i\}_{i=1}^{n}$ the *gene expression data* taken to analyse their functional influence consists of the expression value of every gene C_{ij}—an expression profile—under one condition m_j in a non-explicitly given set of conditions $M = \{m_j\}_{j=1}^{p}$, which grows as we take more measurements.

Under these premises, *co-regulation* refers to the increment or decrement of the expression value in a set of genes brought about by the change in expression value of other genes. At each condition and for each gene, co-regulation results either in *up-regulation*, an increment in expression value, or *down-regulation*, a decrement in expression value, and these changes are expected to reveal functional relations between genes.

This emphasis on up-regulation and down-regulation make gene profile exploration an ideal candidate to be explored by means of $\mathcal{K}$-Formal Concept Analysis, a flavour of Formal Concept Analysis where incidences take value in a multi-valued algebra $\mathcal{K}$ which is an idempotent semifield—an analogue of a field replacing addition with an idempotent law [1,2,3].

In this paper we will undertake the exploration of expression profiles with $(\overline{\mathbb{R}}_{\min,+})$- $(\overline{\mathbb{R}}_{\max,+})$-Formal Concept Analysis with the purpose of researching into functional response patterns. For that purpose, in Section 2 we review data-preprocessing, $\mathcal{K}$-Formal Concept Analysis and lattice-building procedures applied to expression profiles. Next we describe our results in a database of *Arabidopsis thaliana* profiles, and conclude in Section 4 by comparing ours to previous work on using Formal Concept Analysis on gene expression data.

2 Methods and Tools

2.1 Data Preparation

The main problem with expression data is noise: mRNA concentrations profiles are irreproducible from experiment to experiment due to conditions difficult or impossible to control—such as the thermodynamic environment of reactions or individual specimen ontogenesis, respectively. Besides, measurement techniques also introduce their own kind of noise, since they are also based in chemical reactions—hybridization of mRNA with fluorescent markers. For this reason most measurements are repeated a number of times for each condition. Sometimes these measurements are used to obtain variance- and mean-normalised profiles for each condition. Finally, an actual profile for condition m_j is obtained which we gather in a single matrix C_{ij} of *positive numbers* where i runs over genes and j over conditions.

For each experiment, a special kind of profile, called a *control*, may be measured as a reference for other measurements. Controls are adapted to the kind

of experiment and might be the profile of a mix of cells of a whole specimen—to obtain a *body map*—or a particular mix of specific cells under study—for instance, healthy cells to be compared against cancerous cells. Since controls may be extracted from population of specimens grown in controlled conditions, they are expected to be less noisy. In our experiments, we designate a set of measurements for the same condition as controls and coalesce them into their geometrical mean $\bar{c}_i$. This produces a single control at the expense of reducing the set of measurements.

Since both up-regulation and down-regulation of genes occur in gene expression we would like to cater to exploring both. All profiles excepting controls are entry-wise normalized by the control profile and their logarithm[1] taken to make the resulting number range in $[-\infty, \infty]$ $R_{ij} = \log \frac{C_{ij}}{\bar{c}_i}$. Log-quotients of gene expression values are

- $R_{ij} \leq 0$ if g_i is down-regulated by m_j ,
- $R_{ij} \geq 0$ if g_i is up-regulated by m_j , and
- $R_{ij} = 0$ if the control and the condition expression value are equal.

Call the doubly completed set of reals $\overline{\underline{\mathbb{R}}} = \mathbb{R} \cup \{\pm\infty\}$. The reasoning above would suggest using as carrier set for log-quotient values $\overline{\underline{\mathbb{R}}}$ where further:

- $R_{ij} = \log \frac{0}{k} = -\infty, k \neq 0$ when g_i is not expressed at all in m_j ,
- $R_{ij} = \log \frac{k}{0} = \infty$ when g_i is not expressed in the control condition.

With $|G| = n$, $|M| = p$, we collect all expression profiles into a $(\overline{\underline{\mathbb{R}}})$-valued matrix $R \in \overline{\underline{\mathbb{R}}}^{n \times p}$, and call the triple (G, M, R) a *multi-valued formal context*, where $R_{ij} = \lambda$ reads as "the expression value of gene g_i in condition m_j is λ" . The procedure to obtain specific concept lattices from this context is roughly sketched in the next subsection.

2.2 $\mathcal{K}$-Formal Concept Analysis of Expression Data

A generalisation of Formal Concept Analysis called $\mathcal{K}$-Formal Concept Analysis (kFCA) was introduced in [1,2,3] to cater for the notion of a *degree of incidence*, where $\overline{\mathcal{K}}$ is a complete idempotent semifield $\mathcal{K} = \langle K, \oplus, \otimes, \cdot^{-1}, \bot, e, \top \rangle$. This allows the analysis of real-valued incidences by embedding them into a convenient algebra, to be investigated next.

$\mathcal{K}$-Formal Concept Analysis. Complete idempotent semifields are already lattices with $a \wedge b = a \oplus b, a \vee b = a \otimes (a \oplus b)^{-1} \otimes b$. For a complete idempotent semifield a *semimodule* or vector space $\overline{\mathcal{K}}^n = \langle \overline{K}^n, \oplus, \bot_n \rangle$ is an additive monoid with a scalar multiplication inherited from the multiplication in the semifield. A unitary vector e_i in this vector space is $e_i(i) = e$ and $e_i(k) = \bot_{\mathcal{K}}, i \neq k$. Notice that semimodules have an order induced by that of the underlying semiring. In the case of idempotent semifields, this order is compatible with the $\oplus$ operation $x \leq y \Leftrightarrow x \oplus y = y$ turning them into join-semilattices.

[1] All logarithms are base 2 in this paper.

Matrices over completed idempotent semifields $R \in \overline{\mathcal{K}}^{n\times p}$ are linear forms between vector spaces. For the analysis of expression values we call:

- a row vector in $\mathcal{Y} = \overline{\mathcal{K}}^{1\times n}$ a $\mathcal{K}$-*set of genes*,
- a column vector in $\mathcal{X} = \overline{\mathcal{K}}^{p\times 1}$ a $\mathcal{K}$-*set of conditions*,
- a column vector in the range of R, $\mathrm{Im}(R) \subseteq \overline{\mathcal{K}}^{n\times 1}$ a *(gene) expression profile*,
- a row vector in the range of R^{T}, $\mathrm{Im}(R^{\mathrm{T}}) \subseteq \overline{\mathcal{K}}^{1\times p}$ a *condition profile*.

Note that DNA micro-arrays actually obtain a set expression values for a particular condition m_j later transformed into an expression profile $p(m_j) = R \otimes e_j$ (§2.1). However, the *condition profile* for g_i, the vector of its expression values for different conditions $q(g_i) = e_i^{\mathrm{T}} \otimes R$ is seldom considered of interest in analyses.

Consider the context $(G, M, R)_{\overline{\mathcal{K}}}$ and row- and column-vector spaces $\mathcal{Y} \cong \overline{\mathcal{K}}^n$ and $\mathcal{X} \cong \overline{\mathcal{K}}^p$. The bracket $\langle . \mid R \mid . \rangle : \overline{K}^n \times \overline{K}^p \to \overline{K}, \langle y \mid R \mid x \rangle = y \otimes R \otimes x$ between left and right vector spaces over $\overline{\mathcal{K}}$ is proven in [3] to induce a Galois connection $[(\cdot)^+_{R,\varphi}, {}^+_{R,\varphi}(\cdot)] : \overline{\mathcal{K}}^n \leftharpoonup \overline{\mathcal{K}}^p$. Given an invertible $\varphi \in K$, the φ-*polars* are the dually adjoint maps

$$(y)^+_{R,\varphi} = \bigvee \{\, x \in \overline{K}^p \mid \langle y \mid R \mid x \rangle \leq \varphi \,\} \quad {}^+_{R,\varphi}(x) = \bigvee \{\, y \in \overline{K}^n \mid \langle y \mid R \mid x \rangle \leq \varphi \,\} .$$

For row- and column-vectors a and b , the φ-*formal concept* $(a, b)_\varphi$ is a pair such that $(a)^+_{R,\varphi} = b$ and ${}^+_{R,\varphi}(b) = a$ with a the φ-*extent* and b the φ-*intent*. The parameter $\varphi \in K$ is called the *threshold of existence* and it can be proven to describe a *maximum expression value* allowed for pairs $(a, b) \in \overline{\mathcal{K}}^n \times \overline{\mathcal{K}}^p$ to be considered as members of the φ-formal concept set $\mathfrak{B}^\varphi(G, M, R)_{\overline{\mathcal{K}}}$ [3]. As usual, φ-concepts can be ordered by extents or dually by intents

$$(a_1, b_1) \leq (a_2, b_2) \Leftrightarrow a_1 \leq a_2 \Leftrightarrow b_1 \leq^{\mathrm{d}} b_2 \tag{1}$$

and the set of φ-concepts with this order is the φ-concept lattice $\underline{\mathfrak{B}}^\varphi(G, M, R)_{\overline{\mathcal{K}}}$.

A drawback for data mining purposes is that the φ-concept lattice, has a huge number of concepts—infinite, in the typical case—and is hard to visualize. Therefore, we define the *structural (gene expression) lattice* $\underline{\mathfrak{B}}(G, M, I_R^\varphi)$ of the φ-concept lattice as the concept lattice of a binary incidence, I_R^φ , *related to* R *and intended to focus on those concepts below a threshold of existence* φ.

The following is a procedure to *build and explore* a structural lattice:

Step 1 Fix a threshold φ. Compute the closures of the n unitary row vectors of dimension $1 \times n$, $\gamma(e_i) = \left({}^+_{R,\varphi}((e_i)^+_{R,\varphi}), (e_i)^+_{R,\varphi} \right)$ and p unitary column vectors of dimension $p \times 1$, $\mu(e_j) = \left({}^+_{R,\varphi}(e_j), ({}^+_{R,\varphi}(e_j))^+_{R,\varphi} \right)$.

Step 2 Define a binary incidence I_R^φ between genes and conditions associated to those concepts by $g_i I_R^\varphi m_j \Leftrightarrow \gamma(e_i) \leq \mu(e_j)$.

Step 3 Use a standard tool for Formal Concept Analysis—ConExp [5]—to build and visualize the concept lattice $\underline{\mathfrak{B}}\,(G, M, I_R^\varphi)$.

Because the procedure that selects the formal concepts depends on the threshold φ, typically the algorithm above must be carried out a number of times—one for each choice of φ that is deemed interesting—a process we call *lattice exploration*. This allows us to analyse non-boolean expression matrices using several thresholds of existence.

The choice of idempotent semiring. For the case at hand, therefore, a proper choice for $\overline{\mathcal{K}}$ is $\overline{\mathbb{R}}_{\max,+}$ (read "completed max-plus semiring"), actually an *idempotent semifield*:

$$\overline{\mathbb{R}}_{\max,+} = \langle \underline{\overline{\mathbb{R}}}, \max, \dot{+}, -\cdot, -\infty, 0, \infty \rangle$$

This is the completed set of reals with the "max" operation used as addition and normal addition as multiplication, and subtraction as the multiplicative inverse. As noted elsewhere, completed idempotent semifields come in dually ordered pairs[3, §2.2.2]. The order dual of $\overline{\mathbb{R}}_{\max,+}$ is $\overline{\mathbb{R}}_{\min,+}$, the completed min-plus semiring

$$\overline{\mathbb{R}}_{\min,+} = \langle \underline{\overline{\mathbb{R}}}, \min, \dot{+}, -\cdot, \infty, 0, -\infty \rangle$$

Notice that then $\top_{\overline{\mathbb{R}}_{\min,+}} = -\infty, \bot_{\overline{\mathbb{R}}_{\min,+}} = \infty$ and $-\cdot$ is actually a dual order isomorphism between both lattice structures. In this notation we have $-\infty \dot{+} \infty = -\infty$ and $-\infty \dot{+} \infty = \infty$, which solves several issues in dealing with the separately completed dioids. This structure actually carries a complete lattice structure

$$\langle L, \vee, \wedge, \bot, \top \rangle := \langle \underline{\overline{\mathbb{R}}}, \max, \min, -\infty, \infty \rangle \ .$$

Therefore we posit this structure as an appropriate means for modelling increments with respect to an average value.

Exploring down-regulation with $(\overline{\mathbb{R}}_{\max,+})$-Formal Concept Analysis. By taking $\overline{\mathcal{K}} := \overline{\mathbb{R}}_{\max,+}$ and the bracket $\langle y \mid R \mid x \rangle = y \dot{\otimes} R \dot{\otimes} x$ the *polars* are the dually adjoint maps[2]

$$\begin{aligned} (y)^{+}_{R,\varphi} &= (y \dot{\otimes} R) \mathbin{\dot{\setminus}} \varphi & \overset{+}{}_{R,\varphi}(x) &= \varphi \mathbin{\dot{/}} (R \dot{\otimes} x) \\ &= R^{\circledast} \dot{\otimes} y^{\circledast} \dot{\otimes} \varphi & &= \varphi \dot{\otimes} x^{\circledast} \dot{\otimes} R^{\circledast} \end{aligned} \tag{2}$$

Recall that $e = 0$ is the *unit for multiplication in* $\overline{\mathbb{R}}_{\min,+}$. Since $\langle y \mid R \mid x \rangle = \max_{i,j}\{y_i \dot{+} R_{ij} \dot{+} x_j\}$ selects the highest expression value(s) in R_{ij} subject to the weights in y_i and x_j which act as focusing mechanisms, by keeping $y_i = 0 = x_j$ and $\varphi \leq 0$ we concentrate on negative expression values $R_{ij} \leq 0$, that is down-regulated genes in the concepts defined by (2). Hence to find down-regulated genes of (G, M, R) we have to explore $\underline{\mathfrak{B}}^{\varphi}(G, M, R)_{\overline{\mathbb{R}}_{\max,+}}$ with $\varphi \in (-\infty, 0]$.

[2] Notice how the polars are given a closed expression in the *dual idempotent semifield* $\overline{\mathbb{R}}_{\min,+}$.

Exploring up-regulation with $(\mathbb{R}_{\mathbf{min},+})$-Formal Concept Analysis. To cater to up-regulated genes we simply consider matrix R to be part of a the context $\overline{\mathbb{R}}_{\min,+}$-*valued formal context* $(G, M, R)_{\overline{\mathbb{R}}_{\min,+}}$. By taking the bracket $[y \mid R \mid x] = y \dot{\otimes} R \dot{\otimes} x$ the dually adjoint maps over the *dual order* now define a *minimum degree of existence* required for pairs of vectors to be considered ϕ-concepts.

$$\begin{aligned} (y)^{+}_{R,\phi} &= \bigwedge \{\, x \in X \mid [x \mid R \mid y] \geq \phi \,\} & {}^{+}_{R,\phi}(x) &= \bigwedge \{\, y \in Y \mid [x \mid R \mid y] \geq \phi \,\} \\ &= R^{\circledast} \dot{\otimes} y^{\circledast} \dot{\otimes} \phi & &= \phi \dot{\otimes} x^{\circledast} \dot{\otimes} R^{\circledast} \end{aligned} \tag{3}$$

Since $[y \mid R \mid x] = \min_{i,j}\{y_i \dot{+} R_{ij} \dot{+} x_j\}$ selects the lowest expression value(s) in R_{ij} subject to the weights in y_i and x_j—which act as a masking mechanisms—by keeping $y_i = 0 = x_j$ and $\phi \geq 0$ we concentrate on *positive* expression values $R_{ij} \geq 0$, that is up-regulated genes in the concepts defined by (3). Hence to find up-regulated genes of (G, M, R) we have to explore $\underline{\mathfrak{B}}^{\phi}(G, M, R)_{\overline{\mathbb{R}}_{\min,+}}$ with $\phi \in (0, \infty)$.

Note that since the unitary vectors in $\overline{\mathbb{R}}^{n}_{\min,+}$ are $(e_i)^{-1}$, another way of exploring $\underline{\mathfrak{B}}^{\phi}(G, M, R)_{\overline{\mathbb{R}}_{\min,+}}$ with $\phi \in (0, \infty)$ is to explore $\underline{\mathfrak{B}}^{-\phi}(G, M, -R)_{\overline{\mathbb{R}}_{\max,+}}$.

3 Results

3.1 Data Conditioning

We selected transcriptomic data for *A. thaliana* to analyse the behaviour of the root and the shoots in a Selenium-rich environment. The data used for this simulation was downloaded from the NCBI database[3], the same data has been analysed in [6]. The data come from an Affymetrix Arabidopsis ATH1 Genome Array [7] which measures concentration of predefined mRNA sequences in a given biological sample.

We perform this preprocessing with the *Bioconductor* R-package as in [8] which also allows MAS preprocessing. A comparison among different preprocessing types suggests that RMA—also supported by Bioconductor—can provide better results [9], but MAS preprocessing seems to be more widely accepted.

The data has 8 different gene expression profiles:

- root tissues, two control samples: **root1** and **root2**
- root tissues, two samples with Selenium: **rootSe1** and **rootSe2**
- shoot tissues, two control samples: **shoot1** and **shoot2**
- shoot tissues, two samples with Selenium: **shootSe1** and **shootSe2**

Each of these profiles provides the expression value of $|G| = 22\,810$ genes.

The data were preprocessed as described in Section 2.1 to obtain two different contexts:

[3] http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE9311

– *normalised in the mean of the normal root profiles* $\mathbb{K}_r = (G, M_r, R_r)_{\overline{\mathbb{R}}}$. Thus all the gene expression values, for the gene i, will be normalized by $c_i^r = \sqrt{C_{i\mathbf{root1}} \cdot C_{i\mathbf{root2}}}$. The final gene expression will be

$$R_{ij} = \log \frac{C_{ij}}{c_i^r} \tag{4}$$

where $j \in \{\mathbf{shoot1}, \mathbf{shoot2}, \mathbf{rootSe1}, \mathbf{rootSe2}, \mathbf{shootSe1}, \mathbf{shootSe2}\}$ is one of the remaining 6 different profiles after removing **root1** and **root2**.

– *normalised in the mean of the normal shoot profiles* $\mathbb{K}_s = (G, M_s, R_s)_{\overline{\mathbb{R}}}$. As before the gene expression values, for the gene i, are normalized by: $c_i^s = \sqrt{C_{i\mathbf{shoot1}} \cdot C_{i\mathbf{shoot2}}}$. The final gene expression will be

$$R_{ij} = \log \frac{C_{ij}}{c_i^s} \tag{5}$$

where $j \in \{\mathbf{root1}, \mathbf{root2}, \mathbf{rootSe1}, \mathbf{rootSe2}, \mathbf{shootSe1}, \mathbf{shootSe2}\}$ is one of the 6 different profiles remaining after removing **shoot1** and **shoot2**.

The idea is that each of the lattices explored for each of these contexts will shed light on the Selenium-modified analogue of the control, but the other conditions will further identify expression behaviour. As previously said the number of conditions for, say $\mathbb{K}_r$ is reduced to 6: the conditions used to find the control no longer appear, and the other six profiles are normalized by it. Therefore M_r and M_s are different albeit related.

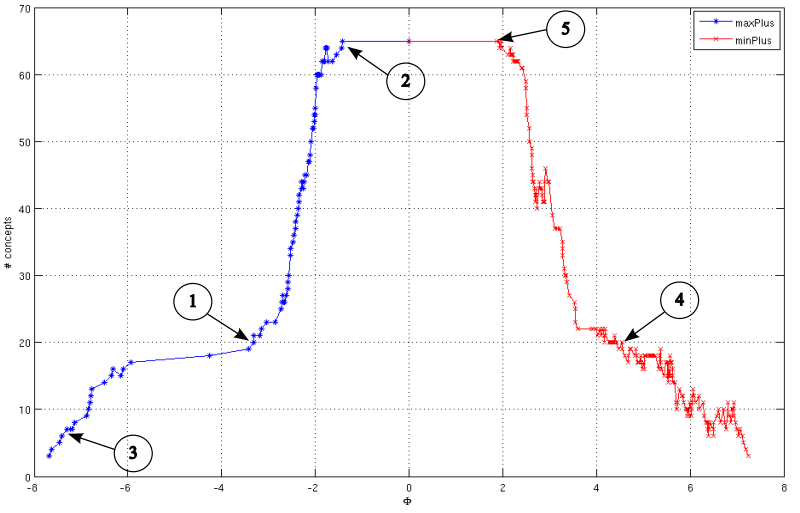
The contexts were processed with our in-house $\mathcal{K}$-Formal Concept Analysis toolbox, running in MatLab.

3.2 Lattice Exploration

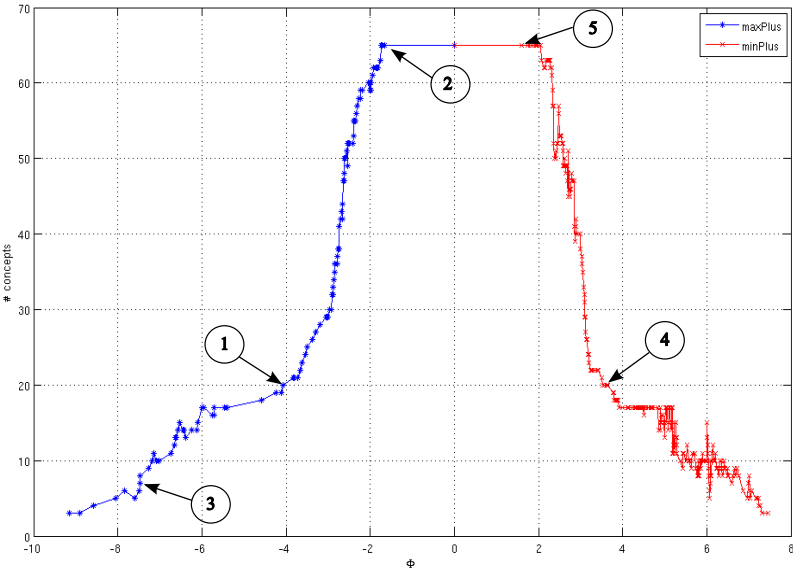
Lattice exploration was carried out on each context using $(\overline{\mathbb{R}}_{\max,+})$- and $(\overline{\mathbb{R}}_{\min,+})$ —to investigate under-expressed and over-expressed genes, respectively—for different values of the thresholds, with φ ranging in $(-\infty, 0)$ and ϕ in $(0, \infty)$, as described in Section 2.2. The resulting number of concepts are shown in Figure 1 for either context.

The overall shape of both curves is very similar. The left halves with $\varphi \in (-\infty, 0)$ start from two concepts when the threshold of existence is below the minimum entry in R, attaining the maximum 2^p in a neighbourhood of 0. On the other hand, the right halves with $\phi \in (0, \infty)$, are roughly symmetric collapsing again into a two-concept lattice when ϕ is above the maximum entry in R . It is worth mentioning, that it is possible to detect a change in the slope of the curves around $\varphi = -6$ and $\phi = 4$. This will be further looked into in the next subsections.

Down-regulation analysis. To obtain an interpretation of the structure of the genes that are down-regulated in the presence of Selenium, structural lattices for negative φ should be explored. Figure 2 depicts two structural lattices at a middle



(a) Root-normalized



(b) Shoot-normalized

Fig. 1. (Colour on-line) Number of concepts as a function of the threshold level $\varphi \in (-\infty, 0)$ for down-regulated genees (blue asterisks) and $\phi \in (0, \infty)$ for up-regulated genes(red crosses) in root-normalized (a) and shoot-normalized (b) data. Points of interest to draw structural lattices from are the leftmost (an example labeled with arrow #3) and rightmost extremes, those points close to the plateaus, coming from either side (examples labeled with arrows #2 and #5), but specially the shoulders are each side of the "mesas" (examples labeled with arrows #1 and #4). The structural lattices for these examples are depicted in the subsequent figures.

value of the left part of the curve where the slope has been found to be lower—a *shoulder*. A clear separation between root-related and shoot-related conditions is appreciated in the form of adjoint sublattices in Figure 2a and almost adjoint sublattices in Figure 2b.

In Figure 2a, the four shoot-related conditions make up a boolean lattice with sets of genes labelled in all possible combinations of the four mentioned conditions. This implies that these conditions cannot be separated at this level in the root normalization. Interestingly, the **RootSe** conditions join at a node with a singleton extent, gene 259161_*at* related to carbon and nitrogen metabolism.

On the other hand, a different situation can be noticed in Figure 2b where the boolean sublattice is now generated by the four root-related conditions while the **ShootSe** conditions are apart. However, they are not so clearly differentiated due to the connection that exists with a lower node of the boolean sublattice. Interestingly, these conditions join at a node with a singleton extent, gene 251196_*at* or *glutaredoxin*, an enzyme normally related to stress signalling which is here inhibited.

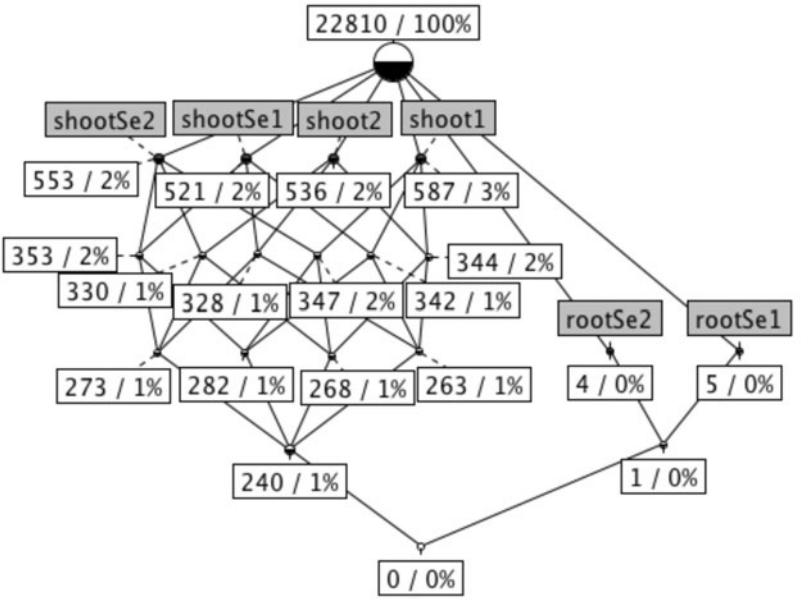
Figure 3 depicts the projection of **rootSe** labels in Figure 3a and **shootSe** labels in Figure 3b from the full boolean lattice of 2^p concepts that appears close to $\varphi = 0$. The bottom nodes represent the 89 (118) genes that are down-regulated by Selenium in the root (the shoots), in which an agreement between both realizations of condition **rootSe** exists. Nonetheless, it is important to acknowledge that at this level of the observation the measurements are not very reliable due to the empirical limitations explained in 2.1, and we will concentrate on the findings for the previous case in Subsection 3.3.

Figure 4 presents, finally, the most salient down-regulated genes in a lattice for a low φ. As can be noted, for the root normalization (resp. shoot) the threshold of existence for **rootSe** (resp. **shootSe**) is too low to allow any gene down-regulated by that condition to appear. However, an incipient structure concerning shoot-related (resp. root-related) conditions is beginning to be discernible which we refuse to analyse in this first attempt.

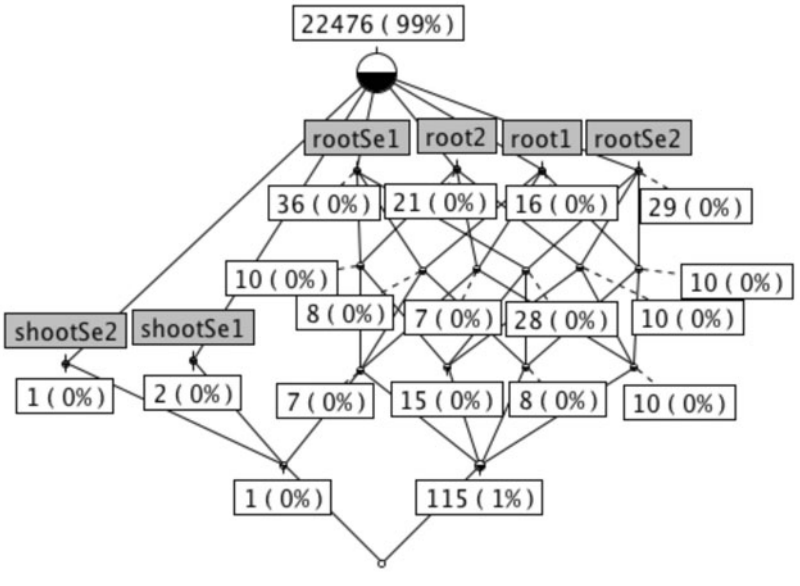
Up-regulation analysis. Changing the choice of semiring from $\overline{\mathbb{R}}_{\max,+}$ to $\overline{\mathbb{R}}_{\min,+}$ allows us to analyse up-regulation. For this case, structural lattices for positive ϕ should be explored.

Figure 5 depicts two structural lattices at both middle values to the right of the curves in Figure 1 where the slopes have been found to be less decreasing. A clear separation between root-related conditions is again evident in the form of adjoint sublattices in Figure 5a. The same cannot be asserted for the shoot-related conditions in Figure 5b as it is not possible to find any structural lattice in which **shootSe1** and **shootSe2** are joined in an independent (not labelled with any other condition) concept different than bottom.

The structure encountered in Figure 5a is analogue to the one in Figure 2a with the four shoot-related conditions conforming a boolean lattice (to the left) and an adjoint sublattice condensing root-related conditions (to the right). The object counts of the concepts are different, however, involving considerably fewer genes in the boolean lattice and many more in the **root** sublattice. As in

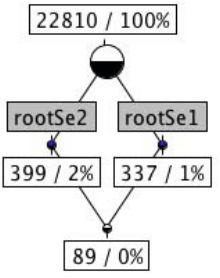


(a) Root-normalized, $\varphi = -3.31$

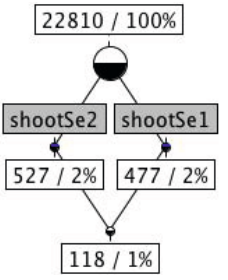


(b) Shoot-normalized, $\varphi = -4.06$

Fig. 2. Structural lattices for down-regulation analysis at a low φ —marked as #1 in both plots of Figure 1—where the **RootSe** (a) and **ShootSe** (b) conditions split away from the rest

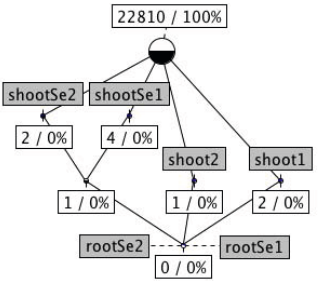


(a) Root-normalized, $\varphi = -1.42$

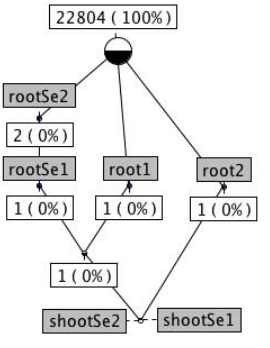


(b) Shoot-normalized, $\varphi = -1.66$

Fig. 3. Structural lattices for down-regulation analysis at a high φ—marked as #2 in either plot of figure 1. Only rootSe (a) and shootSe (b) related conditions are retained, since by considering all conditions at this level a fully connected boolean lattice would be obtained indicating a non-discriminative value of φ.



(a) Root-normalized, $\varphi = -7.29$



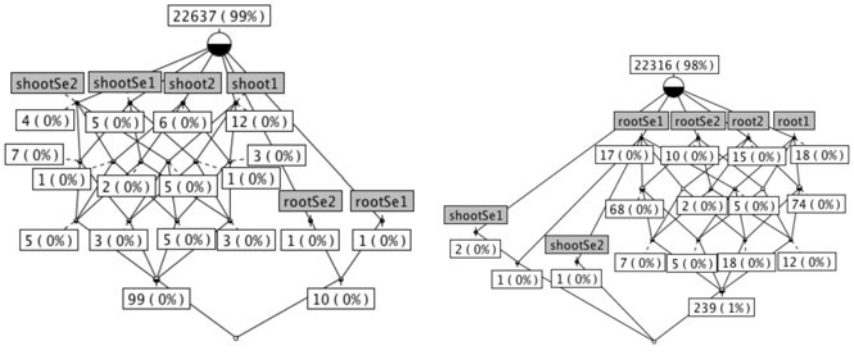
(b) Shoot-normalized, $\varphi = -7.47$

Fig. 4. Structural lattices for down-regulation analysis at a low φ—marked as #3 in either plot of figure 1

down-regulation both **rootSe** conditions join at a node that in this case contains 10 exclusive genes whose analysis can be found in Section 3.3.

Unfortunately, and though almost the inverse situation can be noticed in Figure 5b, where the boolean sublattice is now generated by the four root-related conditions, the **shootSe** conditions do not appear totally apart or even joining at a common concept different to bottom. This divergence between the two realizations of the experiments prevents us from providing findings in this situation.

Figure 6 depicts the projection of **rootSe** labels in Figure 6a and **shootSe** labels in Figure 6b from the full boolean lattice of 2^p concepts that appears close to $\phi = 0$. The bottom nodes represent the 190 (203) genes that are up-regulated by Selenium in the root (shoots, respectively) in which an agreement between both realizations of the experiment exists.



(a) Root-normalized, $\phi = 4.41$ (b) Shoot-normalized, $\phi = 3.65$

Fig. 5. Structural lattices for up-regulation analysis at a ϕ—marked with arrows #4 in both plots of figure 1—where the **RootSe** (a) and **ShootSe** (b) conditions split away from the rest

(a) Root-normalized, $\phi = 1.86$ (b) Shoot-normalized, $\phi = 1.59$

Fig. 6. Structural lattices for up-regulation analysis at a high ϕ—marked as #5 in both plots of figure 1. Only **rootSe** (a) and **shootSe** (b) conditions are retained. With all conditions considered, a fully connected boolean lattice would be obtained at this threshold indicating a non-discriminative value of ϕ .

Finally, similar lattices as the ones depicted for down-regulation in Figure 4 for low φ can be obtained for high ϕ and up-regulation. However, they are omitted here as they do not add much information for the present analysis.

3.3 Findings

We used the gene identifiers appearing in the more reliable concepts, those with lowest down-regulation and highest up-regulation threshold, to obtain their functional description, when available, from a knowledge database.

Preliminary analyses suggest that for up-regulation in the roots subject to *Se*, our procedure detects over-expressed genes used by *A. thaliana* to sense and signal physiological conditions (Ca^{++} transport), to bind to heavy metals (*Cd*, *Zn*) and salts (*Se* is introduced as a selenate) and to combat metal-, pathogen- and salt-induced stress. It is encouraging that one of these genes has an unknown function but is suggested by our procedure to engage in some or all of these functions.

The results for down-regulation are less clear. On the one hand, less genes are clearly under-expressed: for the roots the single reliably-detected gene engages in the metabolism of carbon by non-photosynthetic means and in that of nitrogen. For the shoots, the clearly inhibited gene, glutaredoxin, is an enzyme involved in signalling stress conditions employing sulphur-redox pairs (cysteine). The overall picture is not clear but might suggest that *Se* is interfering with the sensing of *S* in the plant, pretending that sulphur is over-abundant and thereby affecting the signalling related to it.

Further in-depth analysis should be carried out by plant physiologists.

3.4 Summary

The analysis carried out in the previous subsections allows us to reach the following conclusions:

- **root** and **shoot** conditions appear clearly apart in terms of the genes up- or down-regulated in each case. The disparity in the values of C_{ij} observed in them advise a separate analysis which we have implemented by providing two types of normalizations as described in section 3.1.
- Up- and down-regulation can be analysed with the same procedure by changing the carrier semiring in $\mathcal{K}$-Formal Concept Analysis from $\overline{\mathbb{R}}_{\max,+}$ to $\overline{\mathbb{R}}_{\min,+}$. The evolution of the number of concepts in each case proceeds inversely as can be observed in the overall symmetry of Figure 1.
- A consistency of both realizations of the same condition, e.g. **rootSe1** and **rootSe2**, should be always enforced to provide reliability.
- When our focus of attention is the up- or down-regulation in **rootSe** (resp. **shootSe**) conditions, the presence of shoot-related (resp. root-related) ones obscures the analysis, as they appear for very low values of φ (resp. very high values of ϕ), that is, point marked as #3 (resp. #4) in Figure 1).
- Around the value of $\varphi = 0$ at #2 (resp $\phi = 0$ at #5) a full boolean lattice of 2^p concepts appears showing the unreliability of the down-regulation (resp. up-regulation) threshold due to limitations of the measuring technique.
- Finally, a compromise between the two previous situations can be found in the middle of both down and up regulation analysis where figure 1 exhibits a decay of the absolute value of its slope—the *shoulders* of Figure 1. At these positions—marked as #1—**root** and **shoot** conditions separate into two adjoint sublattices for root normalization and a not-so-clear separation for shoot-normalised experiments.
- Pending more thorough analyses, the lattice theory-induced findings can be corroborated by gene-function analysis of the extents found for each case.

4 Discussion

In this paper we have introduced a new approach to gene expression data analysis with $\mathcal{K}$-Formal Concept Analysis, a flavour of Formal Concept Analysis where incidences may take values in complete idempotent semifields. Specifically, we

directly analyse the $\overline{\mathbb{R}}$-valued, non-scaled context of gene expression values by means of $(\overline{\mathbb{R}}_{\min,+})$- and $(\mathbb{R}_{\max,+})$-Formal Concept Analysis.

Our analyses show that a combination of these is a promising tool for the interactive exploration of gene co-regulation, since exploring the context with $(\mathbb{R}_{\max,+})$-Formal Concept Analysis captures the phenomenon of gene down-regulation, while using $(\mathbb{R}_{\min,+})$-Formal Concept Analysis for the exploration captures up-regulation, decreases and increases, respectively, of gene concentrations with respect to a normalizing gene expression profile. In this way, we have detected genes that are either up-regulated or down-regulated in specimens of *A. thaliana* subject to a Selenium-induced physiological stress.

Previous work on using Formal Concept Analysis for transcriptomics includes a remarkable proposal for a methodology for gene expression data exploration in [10], which seems to be the schedule adopted by most practitioners. Pensa et al. suggest and iterative process of exploration based in the *inductive databases* paradigm: for each iteration loop against a database of gene expression data, they carry out pre-processing, data discretisation (attribute scaling), Boolean gene expression data enrichment, Constraint-based extraction of Formal Concepts and post-processing.

Note that our methodology shares the first and last steps, but greatly changes the intermediate steps since no scaling or enrichment is used. Of course, this preliminary work has only demonstrated a single loop of the exploration procedure.

For instance, Motameny et al. [11] concentrated on a binary classification task over human leukaemia. They scaled gene expression values into binary attributes and used standard extents to obtain gene sets inducing rules for classification. In related work, [12] uses *interval scaling* aided by experts to discretise expression values.

Scaling is widely acknowledged to introduce biases in the analysis and perhaps to result in loss of context information [13]. Thresholding and insensibility parameters [14] have been used to minimize these effects, but also richer, hopefully loss-free, kinds of scaling such as *interordinal scaling* [15].

With regard to noise preprocessing, since normalization by means of control conditions does not dispose of noise, practitioners either refuse to trust data too firmly or do a flavour of noise-insensitive analysis [15,14].

Our work is an example of the former: we only describe analysis at thresholds marked as #1 and #4 in Figure 1, far away from values either too close to noise (#2 and #5), or so high in absolute value that they show no significant information (#3).

An instance of the second type of analysis, Pattern Formal Concept Analysis was designed to minimize or dispose of the need for scaling [16] . The novelty in [13,15] is considering expression value intervals as pattern structures to act as "attributes" in the context. The process of lattice building accords narrow intervals to concepts lower in the lattice and wider intervals to those higher up. The wider the interval, the less reliable is the concept association between genes and conditions. This seems to be a complementary approach to our analysis

based in the threshold of existence for concepts, but it has not been applied to the complementary process of gleaning up- and down-regulated genes.

Regarding the phenomena being explored, most of the work so far seems to have concentrated in over-expressed genes or up-regulation, whereas our framework also caters for down-regulation, albeit with a technique complementary to that used for up-regulation, that is $(\overline{\mathbb{R}}_{\min,+})$- vs. $(\overline{\mathbb{R}}_{\max,+})$-Formal Concept Analysis.

In future work we plan to attack control vs. case studies in *A. thaliana*, as well as using the different types of Galois connections of Extended Formal Concept Analysis [3] on gene expression data to widen the array of tools at the practitioner's disposal.

Acknowledgments

We would like to thank Dr. F. Valverde, Research Scientist, Instituto de Bioquímica Vegetal y Fotosíntesis, Consejo Superior de Investigaciones Científicas with his help in interpreting gene extents for *A. thaliana*.

This work has been partially supported by the Spanish Government-Comisión Interministerial de Ciencia y Tecnología projects 2008-06382/TEC and 2008-02473/TEC and the regional project CCG10-UC3M/TIC-5570.

References

1. Valverde-Albacete, F.J., Peláez-Moreno, C.: Towards a generalisation of Formal Concept Analysis for data mining purposes. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3874, pp. 161–176. Springer, Heidelberg (2006)
2. Valverde-Albacete, F.J., Peláez-Moreno, C.: Further Galois connections between semimodules over idempotent semirings. In: Diatta, J., Eklund, P. (eds.) Proceedings of the 4th Conference on Concept Lattices and Applications (CLA 2007), Montpellier, pp. 199–212 (2007)
3. Valverde-Albacete, F.J., Peláez-Moreno, C.: Extending conceptualisation modes for generalised Formal Concept Analysis. Information Sciences (in press)
4. Stoughton, R.: Applications of DNA microarrays in biology. Biochemistry 74, 53 (2005)
5. Yevtushenko, S.A.: System of data analysis "Concept Explorer". In: [17], pp. 127–134 (in Russian), http://sourceforge.net/projects/conexp
6. Van Hoewyk, D., Takahashi, H., Inoue, E., Hess, A., Tamaoki, M., Pilon-Smits, E.A.H.: Transcriptome analyses give insights into Selenium-stress responses and Selenium tolerance mechanisms in *arabidopsis*. Physiologia Plantarum 132, 236–253 (2008)
7. Affymetrix. Statistical algorithms description document, Santa Clara, Ca (2002)
8. Gentleman, R., Carey, V., Huber, W., Irizarry, R., Dudoit, S. (eds.): Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Statistics for Biology and Health. Springer, Heidelberg (2005)
9. Irizarry, R.A.: Summaries of Affymetrix GeneChip probe level data. Nucleic Acids Research 31, 15e–15 (2003)

10. Pensa, R., Besson, J., Boulicaut, J.: A methodology for biologically relevant pattern discovery from gene expression data. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 230–241. Springer, Heidelberg (2004)
11. Motameny, S., Versmold, B., Schmutzler, R.: Formal Concept Analysis for the identification of combinatorial biomarkers in breast cancer. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 229–240. Springer, Heidelberg (2008)
12. Gebert, J., Motameny, S., Faigle, U., Forst, C., Schrader, R.: Identifying genes of gene regulatory networks using Formal Concept Analysis. Journal of Computational Biology 15, 185–194 (2008)
13. Kaytoue, M., Duplessis, S., Kuznetsov, S.O., Napoli, A.: Two FCA-based methods for mining gene expression data. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS, vol. 5548, pp. 251–266. Springer, Heidelberg (2009)
14. Pensa, R., Boulicaut, J.: Towards Fault-Tolerant Formal Concept Analysis. In: Bandini, S., Manzoni, S. (eds.) AI*IA 2005. LNCS (LNAI), vol. 3673, pp. 212–223. Springer, Heidelberg (2005)
15. Kaytoue, M., Kuznetsov, S., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in Formal Concept Analysis. In: Information Sciences (2011)
16. Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 129–142. Springer, Heidelberg (2001)
17. ACM: Proceedings of the 7th National Conference on Artificial Intelligence KII 2000, Russia, ACM (2000)

Biclustering Numerical Data in Formal Concept Analysis

Mehdi Kaytoue[1], Sergei O. Kuznetsov[2], and Amedeo Napoli[1]

[1] Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA)
Campus Scientifique, B.P. 70239 – Vandœuvre-lès-Nancy – France
kaytouem@loria.fr, napoli@loria.fr
[2] State University Higher School of Economics
Pokrovskiy Bd. 11 – 109028 Moscow – Russia
skuznetsov@hse.ru

Abstract. A numerical dataset is usually represented by a table where each entry denotes the value taken by an object in line for an attribute in column. A bicluster in a numerical data table is a subtable with close values different from values outside the subtable. Traditionally, largest biclusters were found by means of methods based on linear algebra. We propose an alternative approach based on concept lattices and lattices of interval pattern structures. In other words, this paper shows how formal concept analysis originally tackles the problem of biclustering and provides interesting perspectives of research.

keywords: biclustering, numerical data, formal concept analysis, pattern structures, conceptual scaling.

1 Introduction

We consider the problem of biclustering numerical data [7,4,16] using techniques of Formal Concept Analysis (FCA) [5,6]. A numerical dataset is given by sets of objects, attributes, and attribute values for objects (many-valued contexts in terms of FCA). The description of an object is a tuple of values, each component corresponding to an attribute value. An example of numerical dataset is given in Table 1 where lines denote objects, while columns denote attributes.

To analyze such a dataset, a major data-mining task is clustering, a data analysis technique used in several domains, e.g. gene expression data analysis. It allows one to group objects into clusters according to some similarity criteria between their description, the similarity being defined according to an adequate distance, following given characteristics [9]. However, clusters are *global* patterns since similarity between objects is computed w.r.t. all attributes simultaneously (possibly weighted). In many applications, and especially in gene expression data analysis, *local* patterns are preferred [3,16] and consist in pairs (A, B) where A is a subset of objects related to a subset of attributes B. Indeed, it is known that a set of genes is activated (e.g. translated into proteins for enabling a biological process) under some conditions only, i.e. only for some attributes. Accordingly,

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 135–150, 2011.

a bicluster is generally represented by a rectangle of values in a numerical data table, see e.g. a bicluster in Table 2. In Table 1, one can see that both biclusters $(\{g_1, g_2\}, \{m_1, m_2, m_3, m_4\})$ and $(\{g_1, g_2\}, \{m_5\})$ give more meaningful information than cluster $\{g_1, g_2\}$ being described by all attributes, since the values taken by objects in A for attributes in B are more similar.

There are many definitions of a bicluster, depending on the relation between subsets of objects and subsets of attributes, as discussed in [16]. In this paper, we consider two types of biclusters: firstly, constant biclusters that can be represented as rectangle of equal values (see Table 3), and secondly, biclusters of similar values, that can be represented by rectangle of similar values (see Table 4). In general case, extracting all biclusters is an intractable problem [16], so in practice heuristics are used. Obviously, even best heuristics may result in the loss of "interesting" biclusters.

The purpose of this paper is to show that an approach based on Formal Concept Analysis (FCA [5]) can be used for biclustering numerical data, leading to a complete, correct and non-redundant enumeration of all maximal biclusters (either of constant or similar values). Such non-heuristic based enumeration has not been deeply considered in the literature due to the very important number of possible biclusters. Whereas a first study is given in [2], we propose here two equivalent FCA-based methods, whose underlying closure operator enables a natural enumeration of maximal biclusters. The first one relies on conceptual scaling (discretization) of numerical data giving rise to several binary tables from which biclusters can be extracted as formal concepts. A second method avoids *a priori* scaling and is based on interval pattern structures [6,12], an FCA formalism that allows one to build concept lattices directly from numerical data from which biclusters of interest can be extracted.

The paper is organized as follows. We first give a brief introduction to FCA, before formally stating the problem of extracting biclusters from numerical data. Then, Section 2 presents the first method based on scaling while Section 3 details the method based on pattern structures. Finally, a discussion compares both approaches w.r.t. their scalability and usage, and highlights several perspectives of research.

1.1 Preliminaries on FCA

We use standard notations of [5]. Let G and M be arbitrary sets and $I \subseteq G \times M$ be an arbitrary binary relation between G and M. The triple (G, M, I) is called a formal context. Each $g \in G$ is interpreted as an object, each $m \in M$ is interpreted as an attribute. The fact $(g, m) \in I$ is interpreted as "g has attribute m". The two following derivation operators $(\cdot)'$ are considered:

$$A' = \{m \in M \mid \forall g \in A : gIm\} \qquad for\ A \subseteq G,$$
$$B' = \{g \in G \mid \forall m \in B : gIm\} \qquad for\ B \subseteq M$$

which define a Galois connection between the powersets of G and M. For $A \subseteq G$, $B \subseteq M$, a pair (A, B) such that $A' = B$ and $B' = A$, is called a *(formal) concept.*

Concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_2 \subseteq B_1)$. With respect to this partial order, the set of all formal concepts forms a complete lattice called the *concept lattice* of the formal context (G, M, I). For a concept (A, B) the set A is called the *extent* and the set B the *intent* of the concept. Certain data are not given directly by binary relations, e.g. numerical data. Such data is usually represented by a many-valued context (G, M, W, I), a 4-tuple constituted of a set of objects G, a set of attributes M, a set of attribute values W and a ternary relation I defined on the Cartesian product $G \times M \times W$. $(g, m, w) \in I$, also written $g(m) = w$, means that "the value of attribute m taken by object g is w". The relation I verifies that $g(m) = w$ and $g(m) = v$ always implies $w = v$. For applying the FCA machinery, a many-valued context needs to be transformed into a formal context with so-called conceptual scaling. The choice of a scale should be wisely done w.r.t. data and goals since affecting the size, the interpretation, and the computation of the resulting concept lattice.

1.2 Problem Setting

Here a numerical dataset is realized by a many-valued context (G, M, W, I) where W is a set of values that objects $g \in G$ can take for attributes $m \in M$. Such many-valued contexts are usually represented by a numerical table where a table-entry gives the value $m(g) \in W$, i.e. the value taken by attribute m in column for object g in line. The Table 1 gives an example (taken from [2]) that we consider throughout this paper, with objects $G = \{g_1, ..., g_4\}$, attributes $M = \{m_1, ..., m_5\}$, and e.g. $m_2(g_4) = 9$.

A bicluster is given by a pair (A, B) with $A \subseteq G$ and $B \subseteq M$. Intuitively, a bicluster is represented by a rectangle of values, or sub-table (modulo line and column permutations), see e.g. the bicluster $(\{g_2, g_3, g_4\}, \{m_3, m_4\})$ highlighted grey in Table 2.

Definition 1 (Bicluster). *Given a numerical dataset (G, M, W, I), a bicluster is a pair (A, B) with $A \subseteq G$ and $B \subseteq M$.*

In [16], several types of biclusters are introduced. The type of a bicluster (A, B) depends on the relation between the values taken by attributes in B for objects in A. In this paper, we consider constant biclusters (equality relation) and biclusters of similar values (similarity relation) as defined in the next paragraphs.

A constant bicluster can be interpreted as a rectangle of identical values, and is defined as follows.

Table 1. A numerical dataset

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	6
g_2	2	1	1	0	6
g_3	2	2	1	7	6
g_4	8	9	2	6	7

Table 2. $(\{g_2, g_3, g_4\}, \{m_3, m_4\})$

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	6
g_2	2	1	**1**	**0**	6
g_3	2	2	**1**	**7**	6
g_4	8	9	**2**	**6**	7

Table 3. A constant bicluster

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	**6**
g_2	2	1	1	0	**6**
g_3	2	2	1	7	**6**
g_4	8	9	2	6	7

Table 4. A bicluster of similar values

	m_1	m_2	m_3	m_4	m_5
g_1	**1**	**2**	**2**	1	6
g_2	**2**	**1**	**1**	0	6
g_3	**2**	**2**	**1**	7	6
g_4	8	9	2	6	7

Definition 2 (Constant bicluster). *Given a numerical dataset* (G, M, W, I)*, a constant bicluster is a bicluster* (A, B) *such that* $m_i(g_j) = m_k(g_l), \forall g_j, g_l \in A, \forall m_i, m_k \in B$.

Since the number of possible biclusters in a numerical dataset can be very large, the notion of maximality gives naturally rise to maximal constant biclusters, i.e. "largest rectangles of identical values".

Definition 3 (Maximal constant biclusters). *Given a numerical dataset* (G, M, W, I)*, a constant bicluster* (A, B) *is maximal if it does not exist a constant bicluster* (E, F) *with either* $A \subset E$ *or* $B \subset F$.
In other terms, (A, B) *is a maximal constant bicluster iff*

- $(A \cup \{g\}, B)$ *is not a constant bicluster* $\forall g \in G \backslash A$
- $(A, B \cup \{m\})$ *is not a constant bicluster* $\forall m \in M \backslash B$

Table 3 shows an example of maximal constant bicluster $(\{g_1, g_2, g_3\}, \{m_5\})$. One should remark that $(\{g_1, g_2\}, \{m_5\})$ is constant but not maximal. Note that maximal constant biclusters taking values 1 in a 1/0 table are formal concepts.

The fact that constant biclusters correspond to sets of objects taking equal values for same attributes is a too strong condition in real-world data. This may lead to the well-known problem of pattern overwhelming. Instead of considering equality, one may relax this condition and consider a similarity relation between values. This idea was introduced in [2] for handling noise in a numerical dataset. Two values $w_1, w_2 \in W$ are said to be similar if their difference does not exceed a user-defined parameter θ. A similarity relation denoted by $\simeq_\theta$ is formally defined by: $w_1 \simeq_\theta w_2 \iff |w_1 - w_2| \leq \theta$. According to this formalization of similarity, a bicluster of similar values can be defined as a "generalization" of constant biclusters.

Definition 4 (Bicluster of similar values). *A bicluster* (A, B) *is a bicluster of similar values if* $m_i(g_j) \simeq_\theta m_k(g_l), \forall g_j, g_l \in A, \forall m_i, m_k \in B$.

Definition 5 (Maximal biclusters of similar values). *A bicluster of similar values* (A, B) *is maximal if there does not exist a bicluster of similar values* (E, F) *with either* $A \subset E$ *or* $B \subset F$.

Table 4 shows an example of maximal bicluster of similar values $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ with $\theta = 1$. Note that bicluster $(\{g_1, g_2\}, \{m_1, m_2\})$ fulfils the

conditions of similarity but is not maximal. Obviously, constant biclusters are biclusters of similar values when $\theta = 0$.

In this paper we consider the problem of mining all maximal (i) constant biclusters and (ii) biclusters of similar values from a numerical dataset. The novelty here lies in the use of Formal Concept Analysis for a correct, complete and non-redundant enumeration (without heuristics). Indeed, we show in the following sections how to define a scaling to build formal contexts whose concepts exactly correspond to the two types of biclusters. However, this leads to the definition of several contexts whose preparation and mining may be inefficient. Then, based on so-called interval pattern structures, we show how binarization can be avoided, which results in reducing practical computational complexity.

2 Mining Biclusters by Means of Conceptual Scaling

In this section, we present two scaling procedures allowing to build formal contexts from which (i) constant biclusters and (ii) biclusters of similar values, can be extracted within the existing FCA framework. Intuitively, scaling allows to express bicluster searchspace under the form of binary tables, while the Galois connection allows to extract maximal biclusters represented as concepts.

2.1 Constant Biclusters

A maximal constant bicluster can be interpreted as a maximal rectangle of identical values. Recall that formal concepts correspond to maximal rectangles of 1 values in binary tables. Accordingly, a maximal constant bicluster containing values $w \in W$ from a numerical dataset (G, M, W, I) corresponds to a concept in a context $\mathbb{K}_w = (G, M, I_w)$ where $(g, m) \in I_w \iff m(g) = w$. One should naturally consider one formal context for each value $w \in W$, which results in a context family $\mathbb{K}_W$ defined as follows:

$$\mathbb{K}_W = \{\mathbb{K}_w = (G, M, I_w) \mid w \in W \ (m, g) \in I_w \iff m(g) = w\}$$

The procedure building the family $\mathbb{K}_W$ from (G, M, W, I) involves one conceptual scaling for each $w \in W$ (actually nominal scalings related to each value w [5]). Figure 1 gives $\mathbb{K}_w = (G, M, I_w)$ for $w = 1$ and $w = 6$. The collection of concepts of each context $\mathbb{K}_w = (G, M, I_w)$ is denoted by $\mathfrak{B}(G, M, I_w)$, or simply $\mathfrak{B}_w$. Examples are given in Figure 1.

The two obvious propositions hold.

Proposition 1. *Given a set of objects $A \subseteq G$ and a set of attributes $B \subseteq M$, a concept (A, B) of $\mathbb{K}_w$ corresponds to a maximal constant bicluster (A, B) of values w from numerical dataset (G, M, W, I).*

Proposition 2. *There is a one-to-one correspondence between the set of concepts $\bigcup_{w \in W} \mathfrak{B}_w$ and the set of all maximal biclusters.*

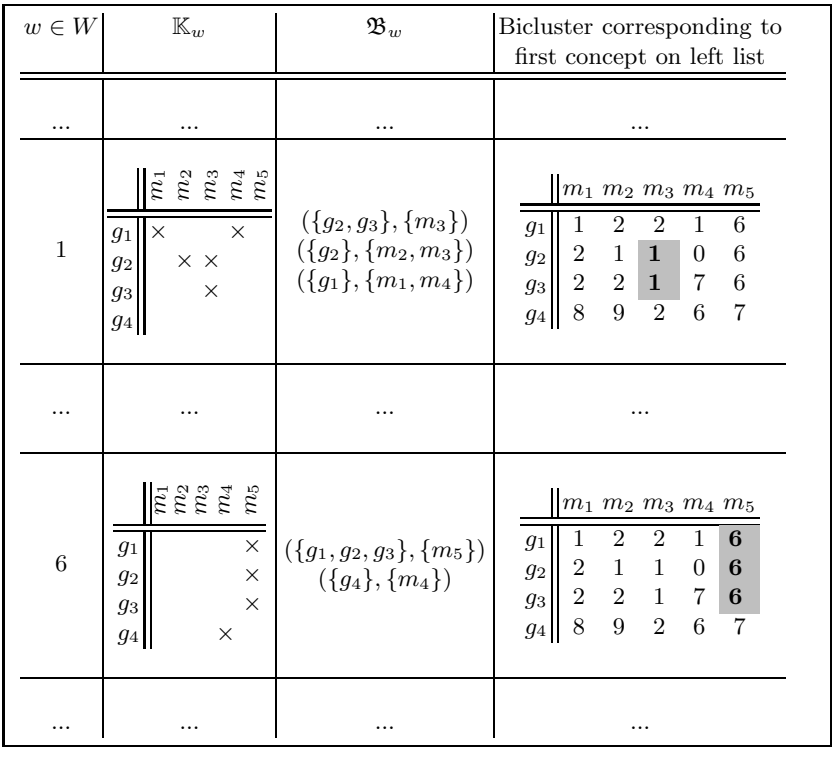


Fig. 1. Extracting constant biclusters from the dataset of Table 1

Hence, an algorithm that constructs the set of concepts $\bigcup_{w \in W} \mathfrak{B}_w$ gives a correct, complete and non redundant enumeration of all maximal constant biclusters.

Figure 1 gives two examples of concepts and their corresponding bicluster representation in the original numerical table.

2.2 Biclusters of Similar Values

The number of constant biclusters can be very large in real-world data, where numerical attribute domains contain many different values. Moreover, it leads to a huge number of artifacts, e.g. the maximal constant bicluster $(A, B) = (\{g_4\}, \{m_4\})$ is a rectangle of area 1, i.e. the product $|A| \times |B|$. One should therefore relax the equality constraint on numerical values when performing scaling with similarity relation $\simeq_\theta$ defined in the introduction. Intuitively, with $\theta = 1$, the previous example is not maximal anymore, whereas $(\{g_3, g_4\}, \{m_4, m_5\})$ is maximal with area equal to 4. For that matter, one should extract rectangles with pairwise similar values w.r.t $\simeq_\theta$. However, this relation is reflexive and symmetric but not transitive, hence a *tolerance relation*.

As related in [14], a tolerance relation T over an arbitrary set G, i.e. $T \subseteq G \times G$, can be represented by a formal context (G, G, T). A formal concept of (G, G, T)

Table 5. Formal context of relation $\simeq_\theta$ over $W = \{0, 1, 2, 5, 6, 7, 8, 9\}$ with $\theta = 1$ (left). Corresponding tolerance classes (middle). Renaming classes as the convex hull of their elements (right).

$\simeq_1$	0	1	2	6	7	8	9
0	×	×					
1	×	×	×				
2		×	×				
6				×	×		
7				×	×	×	
8					×	×	×
9						×	×

Classes of tolerance
$\{0, 1\}$
$\{1, 2\}$
$\{6, 7\}$
$\{7, 8\}$
$\{8, 9\}$

Renamed classes
$[0, 1]$
$[1, 2]$
$[6, 7]$
$[7, 8]$
$[8, 9]$

where intent is equal to extent corresponds to a *class of tolerance*, i.e., a maximal subset of G such that all pairs of its elements are in relation T.

Going back to the tolerance relation $\simeq_\theta$ on a set of values W, tolerance classes are maximal sets of pairwise similar values, corresponding to concepts (A, B) of $(W, W, \simeq_\theta)$ such that $A = B$ [11]. This is exactly what we need to characterize maximal biclusters of similar values. More details on this process are given in [11], while Table 5 shows initial context $(W, W, \simeq_\theta)$ and corresponding classes of tolerance from the numerical dataset of Table 1.

Now that classes of tolerance, or maximal sets of pairwise similar values, are characterized and computed, we can rename them for sake of readability and use them for scaling the initial dataset from which maximal biclusters of similar values can be extracted.

We choose to rename a class $K \subseteq W$ as the convex hull of its elements, i.e. the interval $[k_i, k_j]$ s.t. k_i and k_j are respectively smallest and largest values of K w.r.t. natural order $\leq$ on numbers. Indeed, when $|K|$ becomes large for certain data, this new name is more concise. Moreover, any $k \in [k_i, k_j]$ respects $k \simeq_\theta k_i \simeq_\theta k_j$.

Biclusters of similar values are a generalization of constant ones, i.e. with all values included in interval $[k_i, k_j]$ for a given class of tolerance. We should now also consider one formal context for each class of tolerance, hence a family of contexts. Consider a numerical dataset (G, M, W, I), and a class of tolerance from W which corresponds to the interval $[k_i, k_j]$. The associated formal context is given by:

$$(G, M, I_{[k_i,k_j]}) \; s.t. \; (g, m) \in I \Leftrightarrow m(g) \in [k_i, k_j] \; and \; (\exists h_1, h_2 \in m' \; s.t. \; m(h_1) = k_i \; and \; m(h_2) = k_j \; or \; \exists n_1, n_2 \in g' \; s.t. \; n_1(g) = k_i \; and \; n_2(g) = k_j)$$

First condition $m(g) \in [k_i, k_j]$ means that $m(g)$ should be similar with all elements of the current class of tolerance. The two other conditions come from the fact that classes of tolerance are computed from the set W: since a bicluster is represented by a rectangle in the numerical table, we should consider only

Class of tolerance	Formal context[a]	Concepts	Bicluster corresponding to first concept on left list
$[0, 1]$	$m_2\ m_3\ m_4$ g_1 × g_2 × × ×	$(\{g_1, g_2\}, \{m_4\})$ $(\{g_2\}, \{m_2, m_3, m_4\})$	$m_1\ m_2\ m_3\ m_4\ m_5$ g_1 1 2 2 **1** 6 g_2 2 1 1 **0** 6 g_3 2 2 1 7 6 g_4 8 9 2 6 7
$[1, 2]$	$m_1\ m_2\ m_3\ m_4$ g_1 × × × × g_2 × × × g_3 × × × g_4 ×	$(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ $(\{g_1\}, \{m_1, m_2, m_3, m_4\})$ $(\{g_1, g_2, g_3, g_4\}, \{m_3\})$	$m_1\ m_2\ m_3\ m_4\ m_5$ g_1 **1** **2** **2** 1 6 g_2 **2** **1** **1** 0 6 g_3 **2** **2** **1** 7 6 g_4 8 9 2 6 7
$[6, 7]$	$m_4\ m_5$ g_1 × g_2 × g_3 × × g_4 × ×	$(\{g_3, g_4\}, \{m_4, m_5\})$ $(\{g_1, g_2, g_3, g_4\}, \{m_5\})$	$m_1\ m_2\ m_3\ m_4\ m_5$ g_1 1 2 2 1 6 g_2 2 1 1 0 6 g_3 2 2 1 **7** **6** g_4 8 9 2 **6** **7**
$[7, 8]$	$m_1\ m_5$ g_4 × ×	$(\{g_4\}, \{m_1, m_5\})$	$m_1\ m_2\ m_3\ m_4\ m_5$ g_1 1 2 2 1 6 g_2 2 1 1 0 6 g_3 2 2 1 7 6 g_4 **8** 9 2 6 **7**
$[8, 9]$	$m_1\ m_2$ g_4 × ×	$(\{g_4\}, \{m_1, m_2\})$	$m_1\ m_2\ m_3\ m_4\ m_5$ g_1 1 2 2 1 6 g_2 2 1 1 0 6 g_3 2 2 1 7 6 g_4 **8** **9** 2 6 7

[a] Empty lines and columns are omitted.

Fig. 2. Extracting all maximal biclusters of similar values from Table 1

similar values in column (second condition) or dually lines (third condition) to test whether a value belongs to a class of tolerance.

Consider the formal context $\mathbb{K}_{[k_i,k_j]}$ which corresponds to the class of tolerance $[k_i, k_j]$ and a concept (A, B) from this context. The following propositions hold.

Proposition 3. *(A, B) is a maximal bicluster of similar values.*

Proposition 4. *There is a one-to-one correspondence between the set of concepts from all formal contexts* $\mathbb{K}_{[k_i,k_j]}$ *and the set of all maximal biclusters of similar values.*

Thus, an algorithm computing the set of concepts from all formal contexts $\mathbb{K}_{[k_i,k_j]}$ gives a correct, complete and non redundant enumeration of maximal biclusters of similar values.

Figure 2 gives the formal context $\mathbb{K}_{[k_i,k_j]}$ for each class of tolerance $[k_i, k_j]$, their respective concepts and bicluster representation in the initial numerical Table 1.

3 Mining Biclusters from Pattern Concept Lattice

Until now, we presented how (constant) biclusters (of similar) values can be extracted using standard FCA tools such as scaling and concept extraction algorithms. Since resulting binary tables may be numerous and large (i.e. one for each class of tolerance), we present in this section an approach based on pattern structures. Pattern structures are introduced in [6] and can be thought as a "generalization" of formal contexts to complex data from which a concept lattice can be built without *a priori* scaling. We consider in this section only biclusters of similar values, since being more general than constant ones and more useful for real-world applications.

3.1 Pattern Structures

Formally, let G be a set (interpreted as a set of objects), let $(D, \sqcap)$ be a meet-semilattice (of potential object descriptions) and let $\delta : G \longrightarrow D$ be a mapping. Then $(G, \underline{D}, \delta)$ with $\underline{D} = (D, \sqcap)$ is called a *pattern structure*, and the set $\delta(G) := \{\delta(g) \mid g \in G\}$ generates a complete subsemilattice $(D_\delta, \sqcap)$, of $(D, \sqcap)$. Thus each $X \subseteq \delta(G)$ has an infimum $\sqcap X$ in $(D, \sqcap)$ and $(D_\delta, \sqcap)$ is the set of these infima. Each $(D_\delta, \sqcap)$ has both lower and upper bounds, resp. 0 and 1. Elements of D are called *patterns* and are ordered by subsumption relation $\sqsubseteq$: given $c, d \in D$ one has $c \sqsubseteq d \Longleftrightarrow c \sqcap d = c$.
A pattern structure $(G, \underline{D}, \delta)$ gives rise to the following derivation operators $(\cdot)^\square$:

$$A^\square = \prod_{g \in A} \delta(g) \qquad for\ A \subseteq G,$$

$$d^\square = \{g \in G | d \sqsubseteq \delta(g)\} \qquad for\ d \in D.$$

These operators form a Galois connection between the powerset of G and $(D, \sqsubseteq)$. *Pattern concepts* of $(G, \underline{D}, \delta)$ are pairs of the form (A, d), $A \subseteq G$, $d \in D$, such that $A^\square = d$ and $A = d^\square$. For a pattern concept (A, d) the component d is called a *pattern intent* and is a description of all objects in A, called *pattern extent*. Intuitively, (A, d) is a pattern concept if adding any element to A changes d through $(\cdot)^\square$ operator and equivalently taking $e \supset d$ changes A. Like in case of formal contexts, for a pattern structure $(G, \underline{D}, \delta)$ a pattern $d \in D$ is called *closed* if $d^{\square\square} = d$ and a set of objects $A \subseteq G$ is called *closed* if $A^{\square\square} = A$. Obviously, pattern extents and intents are closed.

3.2 Interval Pattern Structures

In [12], a numerical dataset (G, M, W, I) is represented by a so-called interval pattern structure $(G, (D, \sqcap), \delta)$ where D is a set of interval vectors, the i^{th} dimension giving an interval of values from W for attribute $m_i \in M$. We denote such vectors as *interval patterns*. In Table 1, the description of object g_1 is the interval pattern $\delta(g_1) = \langle [1,1], [2,2], [2,2], [1,1], [6,6] \rangle$. Interval patterns can be represented as $|M|$-hyperrectangles in Euclidean space $\mathbb{R}^{|M|}$, whose sides are parallel to the coordinate axes.

Now we detail how interval patterns are ordered. Consider firstly a single attribute $m \in M$, with value domain $W_m \subseteq W$. Elements of W_m can be ordered within a meet-semi-lattice making them potential object descriptions. Recalling that any $w \in W_m$ can be written as interval $[w, w]$, the infimum $\sqcap$ of two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$ is: $[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)]$, i.e. the largest interval containing them. Indeed, when c and d are intervals, $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$ holds:

$$\begin{array}{rrcl} [a_1,b_1] \sqsubseteq [a_2,b_2] \Leftrightarrow & [a_1,b_1] \sqcap [a_2,b_2] & = & [a_1,b_1] \\ \Leftrightarrow & [min(a_1,a_2), max(b_1,b_2)] & = & [a_1,b_1] \\ \Leftrightarrow & a_1 \leq a_2 \text{ and } b_1 \geq b_2 & & \\ \Leftrightarrow & [a_1,b_1] & \supseteq & [a_2,b_2]. \end{array}$$

As objects are described by several intervals, each one standing for a given attribute, *interval patterns* have been introduced as p-dimensional vector of intervals, with $p = |M|$. Given two interval patterns $e = \langle [a_i, b_i] \rangle_{i \in [1,p]}$ and $f = \langle [c_i, d_i] \rangle_{i \in [1,p]}$ their infimum $\sqcap$ and induced ordering relation $\sqsubseteq$ are given by:

$$\begin{array}{l} e \sqcap f = \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqcap \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ \qquad = \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1,p]} \end{array} \qquad \begin{array}{l} e \sqsubseteq f \Leftrightarrow \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqsubseteq \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ \qquad \Leftrightarrow [a_i, b_i] \sqsubseteq [c_i, d_i], \ \forall i \in [1, p] \end{array}$$

This means that patterns with larger intervals are subsumed by patterns with smaller ones. Hence, one can define a pattern structure $(G, (D, \sqcap), \delta)$ from a numerical dataset (G, M, W, I), where $(D, \sqcap)$ is a meet-semi-lattice of interval patterns. This is deeply detailed in [12]. We illustrate here the Galois connection.

$$\begin{array}{l} \{g_2, g_3\}^{\square} = \delta(g_2) \sqcap \delta(g_3) \\ \qquad = \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle \end{array}$$

$$\begin{array}{l} \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle^{\square} = \{g \in G | \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle \sqsubseteq \delta(g)\} \\ \qquad = \{g_2, g_3\} \end{array}$$

Hence $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle)$ is a pattern concept. The set of all pattern concepts gives rise to a pattern concept lattice, see Figure 3 for our example. Intuitively, $(A_1, d_1) \leq (A_2, d_2)$ means that corresponding hyperrectangle of (A_1, d_1) is included in corresponding hyperrectangle of (A_2, d_2).

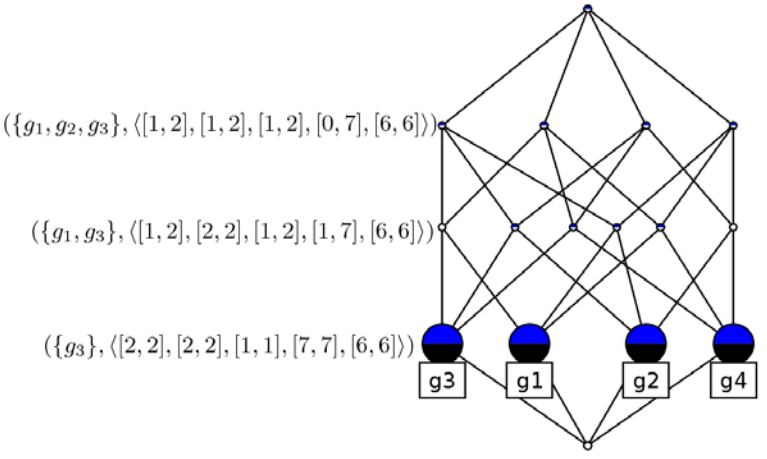


Fig. 3. Pattern concept lattice of pattern structure from Table 1. 3 concepts are fully described with respective pattern extent and intent.

3.3 Biclusters of Similar Values in Pattern Concepts

A pattern concept (A, d) of a numerical dataset (G, W, M, I) can be seen as a bicluster (A, M) since it gives a range of value for each attribute $m \in M$. Bicluster representation of $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle)$ is given in Table 6.

However, a pattern concept (A, d) is not necessarily a bicluster of similar values, for three reasons. First, d may contain intervals larger than θ, i.e. all values in columns are not necessarily similar. Secondly, d may contain different intervals whose values are not similar, i.e. all values in lines may not be similar. Finally, if those conditions are respected, it is not sure that maximality of biclusters holds. We show how to control these statements to extract maximal biclusters of similar values from the pattern concept lattice.

First statement. Avoiding intervals of size larger than θ in a pattern intent d means that a pattern concept will correspond to a rectangle for which each column has similar values. For that matter, consider a modification $(G, (D*, \sqcap), \delta)$ of the interval pattern structure defined in the previous subsection: the set $D*$ consists of tuples, whose components are either intervals or the null element $*$.

Table 6. Interval pattern as bicluster

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	6
g_2	**2**	**1**	**1**	**0**	**6**
g_3	**2**	**2**	**1**	**7**	**6**
g_4	8	9	2	6	7

Table 7. Introducing $\theta = 1$

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	6
g_2	**2**	**1**	**1**	0	**6**
g_3	**2**	**2**	**1**	7	**6**
g_4	8	9	2	6	7

For two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$ their infimum $\sqcap$ is defined as follows: $[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)]$ if $|max(b_1, b_2) - min(a_1, a_2)| \leq \theta$ and $*$ otherwise. Moreover, $* \sqcap [a, b] = *$ for any $a, b \in \mathbb{R}$. Consider that for $d \in D$, d_m denotes the interval given for attribute $m \in M$. Now, given two interval vectors $c = \langle c_i \rangle$ and $d = \langle d_i \rangle$ their infimum is computed componentwise: $c \sqcap d = \langle c_i \sqcap d_i \rangle$. Applying operators of the Galois connection on set $\{g_2, g_3\}$ derives the concept $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], *, [6,6] \rangle)$, while starting with set $\{g_1, g_4\}$ allows to derive concept $(\{g_1, g_2, g_3, g_4\}, \langle *, *, [1,2], *, [6,6] \rangle)$. The resulting pattern concept lattice is given in Figure 4 and contains only 11 concepts compared to 16 when the operation $\sqcap$ is not constrained with θ. Table 7 shows the bicluster representation of $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], *, [6,6] \rangle)$, i.e. a rectangle for which values in each column are similar w.r.t. $\theta = 1$. Note that one should ignore attributes that take the value $*$ in pattern intent.

Second statement. From a pattern structure $(G, (D*, \sqcap), \delta)$, we are able to build a pattern concept lattice whose concepts corresponds to rectangles having similar values in columns. We should therefore also consider similar values in lines. Going back to concept $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], *, [6,6] \rangle)$, we remark that $(\{g_2, g_3\}, \{m_1, m_2, m_3\})$ and $(\{g_2, g_3\}, \{m_5\})$ are biclusters of similar values that can be built from the initial pattern concept. Indeed, the intervals describing attributes m_1, m_2, and m_3 and pairwise similar ($[2,2] \simeq_\theta [1,2] \simeq_\theta [2,2]$ with $\theta = 1$), while interval describing attribute m_5 is similar with no others. We should accordingly consider classes of tolerance between attribute descriptions to extract biclusters of similar values. The similarity relation $\simeq_\theta$ is adapted for intervals as follows: $[a_1, b_1] \simeq_\theta [a_2, b_2] \iff max(b_1, b_2) - min(a_1, a_2) \leq \theta$.

Proposition 5. *Given a pattern concept (A, d), any pair (A, B) with $B \subseteq M$ is a bicluster of similar values iff $\{d_m\}_{\forall m \in B}$ is a class of tolerance w.r.t. relation $\simeq_\theta$ over the set $\{d_m\}_{\forall m \in M}$.*

Proof. Consider that (A, B) is not a bicluster of similar values: $\exists g_1, g_2 \in A$, and $\exists m_1, m_2 \in B$ such that $m_1(g_1) \not\simeq_\theta m_2(g_2)$, a contradiction.

Third statement. By controlling the two first statements, we are able to extract biclusters of similar values from the pattern concept lattice of $(G, (D*, \sqcap), \delta)$. By the properties of classes of tolerance making a class a maximal set of similar values, we know that biclusters are maximal in colums, i.e. no columns can be added without violating the similarity relation. However, we are not sure that biclusters are maximal in lines. Going back to previous example, i.e. $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], *, [6,6] \rangle)$, the extracted biclusters $(\{g_2, g_3\}, \{m_1, m_2, m_3\})$ and $(\{g_2, g_3\}, \{m_5\})$ are not maximal. Indeed, we have $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ and $(\{g_1, g_2, g_3\}, \{m_5\})$ that are also biclusters of similar values. If such biclusters are not maximal, this means that objects can be added in the extent A while B remains the same set. Due to the generalization/specialization property of concept lattices, such larger bicluster can be found in the direct upper neighbours of concept $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], *, [6,6] \rangle)$, i.e. concept $(\{g_1, g_2, g_3\}, \langle [1,2], [1,2], [1,2], *, [6,6] \rangle)$.

Example. The Figure 4 gives the pattern concept lattice of $(G, (D*, \sqcap), \delta)$ with $\theta = 1$. For each pattern intent, elements of each class of tolerance are either underlined, crossed-off, or in bold. For a pattern concept (A, d), when a class is underlined, or in bold, it means that (A, B), B being the set of attribute corresponding to this class, is a maximal bicluster of similar values. If element of the class are crossed-off, this means that (A, B) is not maximal, i.e (C, B) with $A \subset C$ can be characterized also in a direct upper concept. For example, take concept $(\{g_1, g_2\}, \langle [1,2], [1,2], [1,2], [0,1], [6,6] \rangle)$. From this concept, according to classes of tolerance, one can characterize the following biclusters of similar values $(\{g_1, g_2\}, \{m_1, m_2, m_3\})$, $(\{g_1, g_2\}, \{m_4\})$ and $(\{g_1, g_2\}, \{m_5\})$. However, $(\{g_1, g_2\}, \{m_4\})$ is the one only that is maximal, i.e. that cannot be characterized from upper pattern concepts with larger extents.

Hence, all biclusters of similar values can be computed from pattern concepts by standard algorithms. These considerations lead to two dual ways of constructing maximal biclusters of similar values as pattern concepts: bottom-up and top-down.

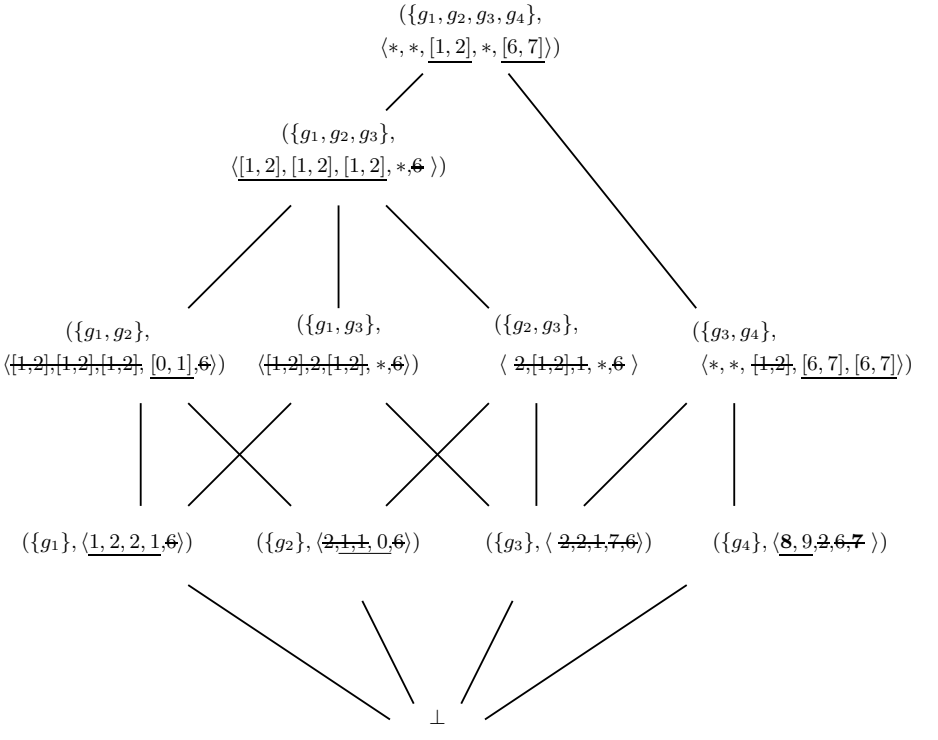


Fig. 4. Pattern concept lattice of pattern structure from Table 1 with $\theta = 1$. When an interval from a pattern intent has same left and right borders, a value is given instead for sake of readability.

4 Discussion and Conclusion

This paper focused on the problem of biclustering numerical data with formal concept analysis. The goal was not to propose a new kind of bicluster, but rather to argue that two existing types of biclusters can be extracted using FCA techniques. For that matter, we proposed two methods producing equivalent results. The first is based on conceptual scaling, while the second on interval pattern structures. It is now expected to experiment these approaches, compare them with other biclustering algorithms (e.g. from [2]) and investigate how to handle other types of biclusters defined in [16]. We should also study the impact of the variation of θ on the concept lattice granularity, or dually on the number of formal contexts/concepts. Finally, we should examine how formal concept analysis in fuzzy seetings can contribute to biclustering problems. Indeed, similarity and tolerance relations are widely studied in such settings [1].

We discuss now our both methods.

Consider the method based on scaling. The strength of such approach is to produce binary tables. Any FCA algorithm (discussed and compared in [15]), or closed itemset algorithm (e.g. Charm [8]) can be used for extracting biclusters. Moreover, since each context of the produced family is independent from the others, a distributed computation is naturally possible: one core can be assigned for each formal context. It also allows to mine other kinds of binary patterns. For example, one can mine fault-tolerant patterns that would correspond to quasi biclusters of similar values, i.e. accepting some exceptions, see e.g. [17]. Meanwhile, searching for frequent biclusters (i.e. involving a number of objects higher than a user-defined threshold [18]) is straightforward. It rises also interesting questions: what is the meaning of an association rule? of a minimal generator?

The second method proposes to extract biclusters from a concept lattice, providing an interesting ordered hierarchy of biclusters. Computing the pattern concept lattice by adapting standard FCA algorithms such as CloseByOne is efficient as experimented in [12], while this algorithm can be parallelized [13]. In [10], CloseByOne was adapted to mine frequent closed interval patterns and their minimal generators. How this algorithm can be adapted for mining frequent biclusters is an interesting perspective of research. The fact that biclusters can be extracted from an ordered hierarchy of concepts make the pattern concept lattice a good structure for user queries. For example, a biologist may be interested in a particular set of genes for a given study. Accordingly, navigating in the concept lattice helps him discovering the different biclusters in which those genes occurs with other good candidates. We can describe such query as extensional since it starts by given a set of objects. On another hand, the approach based on scaling is more useful for so called intentional queries: the biologist is interested in all biclusters with values in a given interval (or class of tolerance) and accordingly only selects the formal context associated to this class.

Acknowledgments

The second author was supported by the project of the Russian Foundation for Basic Research, grant no. 08-07-92497-NTsNIL_a. Other authors were supported by the "Contrat de Plan Etat Région – Modélisation des biomolécules et leurs intéractions" (Lorraine, France, 2007–2013). Authors would like to thank Sébastien Vaillant for implementing the scaling procedure.

References

1. Belohlávek, R., Funioková, T.: Similarity and fuzzy tolerance spaces. J. Log. Comput. 14(6), 827–855 (2004)
2. Besson, J., Robardet, C., Raedt, L.D., Boulicaut, J.F.: Mining bi-sets in numerical data. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 11–23. Springer, Heidelberg (2007)
3. Boulicaut, J.F., Besson, J.: Actionability and formal concepts: A data mining perspective. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 14–31. Springer, Heidelberg (2008)
4. Cheng, Y., Church, G.: Biclustering of expression data. In: Proc. 8th International Conference on Intelligent Systems for Molecular Biology (ISBM), pp. 93–103 (2000)
5. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)
6. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 129–142. Springer, Heidelberg (2001)
7. Hartigan, J.A.: Direct clustering of a data matrix. J. Am. Statistical Assoc. 67(337), 123–129 (1972)
8. Hsiao, C.J., Zaki, M.J.: Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure. IEEE Trans. on Knowl. and Data Eng. 17(4), 462–478 (2005)
9. Jiang, D., Tang, C., Zhang, A.: Cluster analysis for gene expression data: a survey. IEEE Transactions on Knowledge and Data Engineering 16(11), 1370–1386 (2004)
10. Kaytoue, M., Kuznetsov, S.O., Napoli, A.: Pattern Mining in Numerical Data: Extracting Closed Patterns and their Generators. Research Report RR-7416, INRIA (2010)
11. Kaytoue, M., Assaghir, Z., Napoli, A., Kuznetsov, S.O.: Embedding tolerance relations in formal concept analysis: an application in information fusion. In: Huang, J., Koudas, N., Jones, G., Wu, X., Collins-Thompson, K., An, A. (eds.) CIKM, pp. 1689–1692. ACM, New York (2010)
12. Kaytoue, M., Kuznetsov, S.O., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in formal concept analysis. Information Sciences (2010) (in Press, Corrected Proof)
13. Krajca, P., Outrata, J., Vychodil, V.: Advances in algorithms based on cbo. In: Kryszkiewicz, M., Obiedkov, S. (eds.) International Conference on Concept Lattices and Their Applications (2010)
14. Kuznetsov, S.O.: Galois connections in data analysis: Contributions from the soviet era and modern russian research. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 196–225. Springer, Heidelberg (2005)

15. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell. 14(2-3), 189–216 (2002)
16. Madeira, S., Oliveira, A.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics 1(1), 24–45 (2004)
17. Pensa, R.G., Boulicaut, J.F.: Towards fault-tolerant formal concept analysis. In: AI*IA. pp. 212–223 (2005)
18. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with titanic. Data Knowl. Eng. 42(2), 189–222 (2002)

Object Configuration Browsing in Relational Databases

Jens Kötters

Monash University, Melbourne, Australia

Abstract. Conventional means of querying a database for relevant information require some degree of knowledge about the nature and the structural representation of such information. The paper addresses the case where sufficient knowledge is not readily available a priori. A method for data exploration based on the refinement of graphs, which represent summarized views of the underlying data, is proposed and illustrated by a small example featuring relational data extracted from an Electronic Health Record. The method is based on Formal Concept Analysis, extended to the case of several many-valued formal contexts, one for each database table, with additional referential attributes.

1 Introduction

Imagine there is a database likely containing some information somebody is interested in. In order to get the information, one has to know the specific question that is answered by the information, and how to formulate this question as a query in SQL or whatever query language is given. But there are cases where one can start with a general question only, and where more specific questions depend on the answer to previous questions; or where one merely wants an overview of what information is available on a particular subject. A doctor making a general enquiry about a patient would be an example for such a case.

This example is used throughout the paper to introduce a graph-based approach for browsing configurations of semantically related objects in a database. The graphs, called Query Graphs, can be imagined as visualized queries in a point-and-click interface, but they have an exact formal representation based on Formal Concept Analysis (FCA) [12]. In [3], a system based on QuOnto[1] is presented that allows a user to specify graphs which represent queries to a database. Database access is mediated through an ontology encoded by a Description Logic, which allows high-level queries to be formulated. The semantics of these graphs is defined in a way analogous to that which is defined for query graphs in Sect. 3. The difference of our approach is that we use FCA for navigation, which provides a formal background for the task at hand, and allows fine-grained control of query refinement options by using a lattice as the underlying data structure.

An early paper describing the use of FCA for navigation is [13], more dedicated approaches have been developed in the following (see e.g. [7],[9]). Some FCA related literature deals with navigation in databases([20],[8],[2]) or other

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 151–166, 2011.

relational data([11]), but usually these approaches define only a single set of objects to be browsed. However, a recent paper by Ferré[10] features an application similar to the one proposed here (see Sect. 7). Further FCA papers dealing with databases are [16],[17],[14],[18].

Huchard et al. write in [15]:

> When processing [...] complex datasets, it is of prime importance for an analysis tool to hold as much as possible to the initial format so that the semantics is preserved and the interpretation of the final results eased.

Consequently, the authors of [15] define a Relational Context Family(RCF), which describes interrelated objects of different types, and they describe an extension of FCA on top of this description, which preserves the distinction between objects of different types. The statement above appears to be motivated by some application in data analysis, but the paper proceeds with a sample application of UML model transformation – in any case, we believe that a similar statement would also hold for the case of querying and navigation, which is the subject of the paper at hand.

The Query Graphs presented in this paper represent descriptions of several objects, each of a particular type (i.e. data table) represented by a vertex, which are linked through object references (i.e. foreign keys) represented by directed edges. This means that the schema of the database is reflected in the query graphs. The database used for the example has six tables, shown in Fig. 1. The tables are supposed to be the output of some Information Extraction (IE) process, running on the following extract from the "CLEF corpus" presented in [19]:

> The patient has had a lymph node biopsy which shows melanoma in his right groin. It is clearly secondaries from the melanoma on his right second toe. Although his PET scan is normal, he does need a groin dissection. We will perform a CT scan to look at the left pelvic side wall.

The stated purpose of the corpus is the training and evaluation of IE systems for certain kinds of clinical text. Figure 2 shows a modified version of an extraction schema presented in [19] that is used as a database schema here. The tables in Fig. 1 have been created manually by the author, based on the schema and the text extract.

In the following, standard FCA notation and terminology as covered in chapter 1 of the standard textbook [12] will be used. The reader is also supposed to be familiar with the process of scaling a many-valued context. The definitions of "Linked Context Family" and "Query Graph" in the following sections have been chosen to resemble existing definitions of "Relational Context Family" [15], "Abstract Concept Graph" [21] and "Basic Conceptual Graph" [5], so that it is easier to recognize commonalities and differences. A more detailed discussion follows in Sect. 7.

PATIENT	name	date of birth	sex
#P1	xy		male

LOCUS	class	laterality
#L1	lymph node	
#L2	groin	right
#L3	second toe	right
#L4	pelvis	

CONDITION	has location	class	patient
#C1	#L2	melanoma	#P1
#C2	#L3	melanoma	#P1

INVESTIGATION	patient	has finding(C)	has finding(R)	has target	class	date
#INV1	#P1	#C1		#L1	biopsy	
#INV2	#P1		#R1		PET scan	
#INV3	#P1			#L4	CT scan	scheduled

RESULT	evaluation	blood stats...
#R1	normal	

INTERVENTION	has indication	has target	class	date
#INT1		#L2	dissection	scheduled

Fig. 1. Sample database

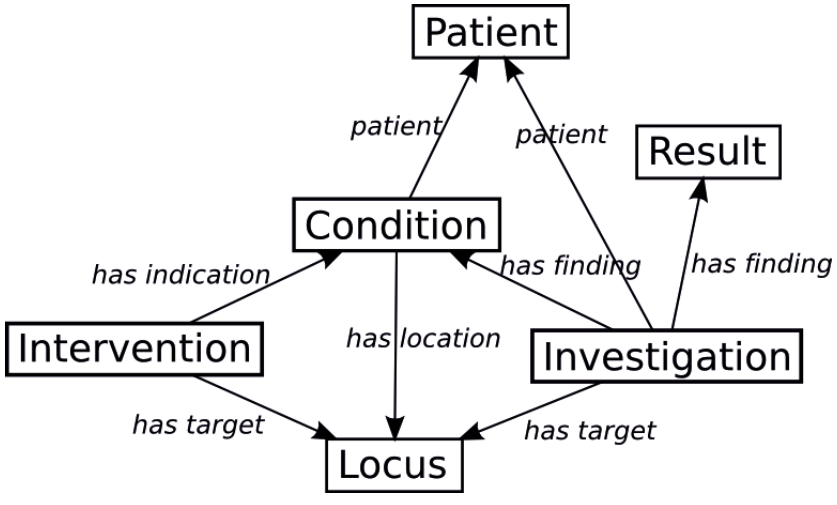


Fig. 2. Data schema from [19] (modified)

2 Linked Context Family

Let T be a set of typenames and $\mathbb{K}_t =: (G_t, M_t, I_t)$ a formal context for $t \in T$. Let L be a set with two maps $\text{src} : L \mapsto T$ and $\text{dest} : L \mapsto T$ on it, such that every $\lambda \in L$ is a partial map from $G_{\text{src}(\lambda)}$ to $G_{\text{dest}(\lambda)}$. We shall call the elements of L *links*, and the pair $((\mathbb{K}_t)_{t \in T}, L)$ a *Linked Context Family(LCF)*.

We will now see how the content of a database is translated into an LCF. The elements of T represent the tables of the database, so for the database in Fig. 1, the elements of T are: CONDITION, INTERVENTION, INVESTIGATION, LOCUS, PATIENT and RESULT. We make the assuption that every table has an attribute "id" which serves as the primary key. For each table $t \in T$, the contents of the id column are collected in a set G_t. We refer to the elements of G_t as objects. In Fig. 1, the id column is always the leftmost column of a table. Those entries which are objects are prefixed by a hash mark (#).

Another assumption is that foreign keys do not span multiple columns. A foreign key always refers to a primary key in one of the tables. However, entries in a foreign key column may be null. This allows to model each foreign key as a partial function $\lambda : G_s \to G_t$, and we set $\mathrm{src}(\lambda) = s$ and $\mathrm{dest}(\lambda) = t$. The case $s = t$ is allowed. For example, the fifth column of the INVESTIGATION table is the function "has target" with src(has_target) = INVESTIGATION and dest(has_target) = LOCUS, defined by has_target(#INV1) = #L1 and has_target(#INV3) = #L4 (the second row contains a null value).

So there are eight links in L, corresponding to the foreign keys in our database. These are represented by the arrows in Fig. 2. Note that some arrows are labeled by the same name, so it may be necessary to modify these names for disambiguation (as for the functions "has_finding" in the INVESTIGATION table).

To obtain the formal context $\mathbb{K}_t$ for a given $t \in T$, we remove the foreign key columns from the table t and consider the remaining table as a many-valued context with object set G_t from which a formal context can be derived by conceptual or logical scaling, where SQL expressions could already be used as the scale attributes (see [16] and [17]). In this paper, we use simple formulas as scale attributes and translate them into SQL later (see Sect. 6).

Consider as an example the INVESTIGATION table. After removing the foreign key columns, we obtain a context with the many-valued attributes "class" and "date". From each many-valued context, a derived context is obtained by scaling, using one scale per attribute. Each scale attribute is expressed as a formula in a single variable X, see the scale for the "class" attribute of the INVESTIGATION table in Fig. 3. The scale encodes a hierarchy of concept names, which can be seen in the line diagram in Fig. 3. For simplicity, we choose nominal scales for all other many-valued attributes in the database. Of course, in a real application it would be desirable that the other scales represent similar hierarchies.

We obtain the attributes of the derived context $\mathbb{K}_t$ by substituting $a(X)$ for X in each scale attribute, for all many-valued attributes a of the corresponding many-valued context. For an example, see the derived context for the INVESTIGATION type in Fig. 4.

This shows by example how a relational database is translated into an LCF. Note that there is generally no canonical translation of a database into an LCF because the scales encode knowledge external to the database, and the construction of suitable scales for a given database may be a nontrivial effort.

class	X ≤ biopsy	X ≤ scan	X ≤ CT scan	X ≤ PET scan
biopsy	×			
scan		×		
CT scan		×	×	
PET scan		×		×

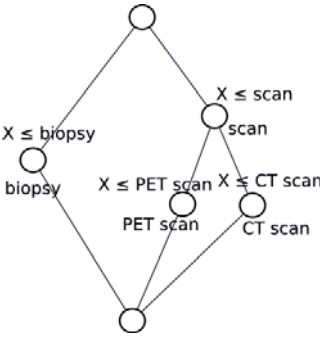


Fig. 3. Scale for class attribute of INVESTIGATION

INVESTIGATION	class(X) ≤ biopsy	class(X) ≤ scan	class(X) ≤ CT scan	class(X) ≤ PET scan	date(X) = scheduled
#INV1	×				
#INV2		×		×	
#INV3		×	×		×

Fig. 4. Derived context for the INVESTIGATION type

3 Query Graphs

Let $(Var_t)_{t\in T}$ be a family of sufficiently large and mutually disjoint sets of variables. Let $Var := \bigcup_{t\in T} Var_t$. We say that a variable $x \in Var_t$ has type t and denote this by writing $\text{type}(x) = t$.

A *query graph* over a given LCF $((\mathbb{K}_t)_{t\in T}, L)$ is a triple (V, E, κ) where

1. $V \subseteq Var$ is a finite set of vertices,
2. $E \subseteq V \times L \times V$ is a finite set of labeled edges (we say that an edge (x, λ, y) goes from x to y and has label λ),
3. the graph is *well-formed*, i.e. $\text{type}(x) = \text{src}(\lambda)$ and $\text{type}(y) = \text{dest}(\lambda)$ for all $(x, \lambda, y) \in E$,
4. κ is a map that assigns a formal concept to each vertex: $\kappa(x) \in \underline{\mathcal{B}}(\mathbb{K}_{\text{type}(x)})$ for all $x \in V$.

In addition we shall require that a query graph must be connected.

Let us call the graph $(T, \{(\mathrm{src}(\lambda), \lambda, \mathrm{dest}(\lambda)) \mid \lambda \in L\})$ the *schema graph* for given T and L. For our example, this is the graph depicted in Fig. 2. It may be interesting to note that the well-formedness property of a query graph means that type is a label-preserving homomorphism from the query graph to the schema graph.

A *realization* of such a query graph is a map $\rho : V \mapsto \bigcup_{t \in T} G_t$ with $\rho(x) \in \mathrm{ext}(\kappa(x))$ for all $v \in V$, where $\mathrm{ext}(\kappa(x))$ denotes the extent of the concept $\kappa(x)$, and furthermore $\lambda(\rho(x)) = \rho(y)$ for all $(x, \lambda, y) \in E$. We denote the set of all realizations of a query graph Q by $P(Q)$.

A morphism preorder can be defined on the query graphs. In [5, pg.30-31,35-38], such a preorder is described for basic conceptual graphs, and we will adapt the definitions and restate the observations made there for the case of query graphs. A map $\varphi : V_1 \to V_2$ from a query graph $Q_1 =: (V_1, E_1, \kappa_1)$ to a query graph $Q_2 =: (V_2, E_2, \kappa_2)$ is called a *morphism* of query graphs if $\mathrm{type}(x) = \mathrm{type}(\varphi(x))$ and $\kappa_1(x) \geq \kappa_2(\varphi(x))$ holds for all $x \in V_1$, and if in addition $(x, \lambda, y) \in E_1 \Rightarrow (\varphi(x), \lambda, \varphi(y)) \in E_2$ holds for all $x, y \in V_1$ and $\lambda \in L$. We say that Q_2 is a refinement of Q_1, if such a morphism exists and denote this by $Q_1 \gtrsim Q_2$. We say that Q_1 and Q_2 are equivalent, in terms: $Q_1 \sim Q_2$, if $Q_1 \gtrsim Q_2$ and $Q_2 \gtrsim Q_1$. Among all query graphs of the same equivalence class, there is a unique graph – up to isomorphism – with a minimal number of vertices. In [5], this graph is called irredundant. We shall use the term *unretractive* instead, because query graphs form a category, and the vertex-minimal graphs of each equivalence class are precisely those graphs which do not allow a retraction onto another (nonisomorphic) graph. The unretractive graphs suggest themselves as representatives of their equivalence classes, and we conclude that the morphism preorder induces a partial order on the set of unretractive query graphs.

Let us finally observe that, if Q_2 is a refinement of Q_1 by virtue of a morphism $\varphi : Q_1 \to Q_2$, the implication $\rho \in P(Q_2) \Rightarrow \rho \circ \varphi \in P(Q_1)$ holds. This means that every realization of Q_2, restricted to the image of Q_1 under φ, can be considered a realization of Q_1, as well. The condition can be strengthened as follows: We shall call Q_2 an *extension* of Q_1 if there is a morphism $\varphi : Q_1 \to Q_2$ and if in addition, every realization in $P(Q_1)$ is of the form $\rho \circ \varphi$ for some $\rho \in P(Q_2)$, i.e. if every realization of Q_1 can be extended to a realization of the refined graph Q_2. In the next section we will get back to the example and consider extensions as part of the result for a query, because they add more information to the graph without precluding any realizations.

4 Navigation Example

Consider the case where a doctor wants to know what information on a particular patient xy is available. The corresponding query is represented by the graph Q_1 in Fig. 5. The graph consists of a single vertex X_P of type PATIENT (indices are used here to indicate the type of variable). The formal concept $\kappa(X_\mathrm{P})$ is stated in terms of its intent, which is the formula "$\mathrm{name}(X_\mathrm{P}) = \mathrm{xy}$" (the variable X_P is used instead of the placeholder variable X initially used, as in Fig. 4). The graph Q_2 in Fig. 5 is the result of the query Q_1. The map $\iota : Q_1 \to Q_2$, $\iota(X_\mathrm{P}) = X_\mathrm{P}$,

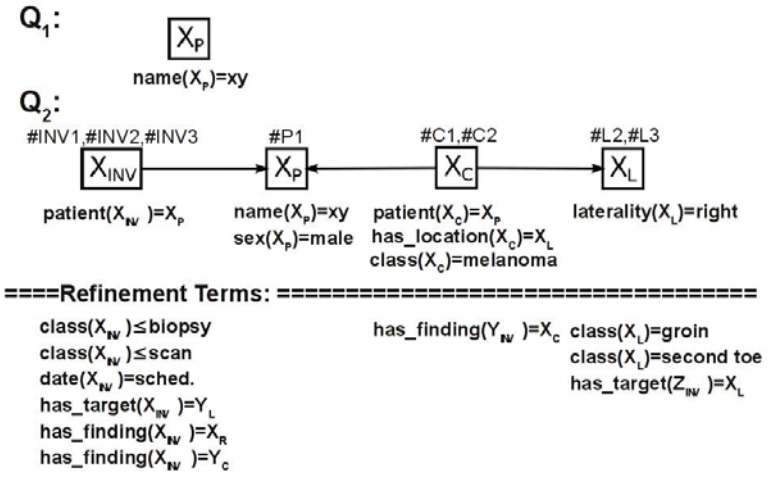


Fig. 5. Navigation example

is a morphism, this verifies that Q_2 is a refinement of Q_1. It is obvious that Q_1 has only a single realization r, given by $r(X_\mathrm{P}) = \#\mathrm{P1}$.

The graph Q_2 has six realizations, which can not be read off directly from the representation in Fig. 5. The objects written on top of the vertices are the *extents* of the vertices, defined by

$$\mathrm{ext}(x) := \{\rho(x) \mid \rho \in P(Q)\}$$

for an arbitrary vertex x of a query graph Q. But all realizations of $P(Q_2)$ map X_P to #P1, in particular there exists a $\rho \in P(Q_2)$ such that $r = \rho \circ \iota$. This verifies that Q_2 is an extension of Q_1.

The most precise concept that describes the extent of a vertex x is given by $\kappa(x) := (\mathrm{ext}(x)'', \mathrm{ext}(x)')$, where $'$ is the derivation operator of the context $\mathbb{K}_{\mathrm{type}(x)}$. The intent of this concept is written below each vertex and, in the case of Q_2, provides the following information: There is exactly one patient with name xy, which is the patient #P1. The patient is male, melanoma have been found on the right side of his body, and three investigations are associated with the patient.

There are two kinds of query refinements proposed by the system, and these can be determined independently for each of the vertices. One kind of refinement is a link in which some, but not all objects in a vertice's extent participate. The other kind is a refinement on the scale values, there is one set of refinements for each of the scaled attributes. A look at the refinement terms for the X_INV vertex of Q_2 reveals that targets and findings of some investigations are available; furthermore, at least one investigation is scheduled, at least one investigation is a scan and at least one investigation is a biopsy. The refinement term below the X_C vertex proposes a link to the investigation which led to the discovery of

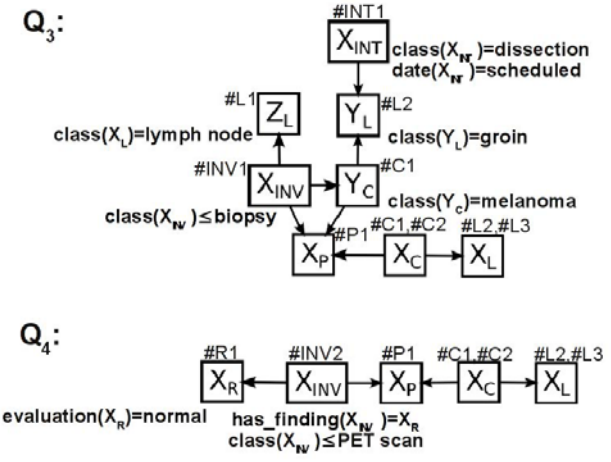


Fig. 6. Navigation example (continued)

the melanoma, and the refinement terms below the X_{L} vertex suggest to have a look at the groin or the second toe, and another investigation link is proposed. Let us say the doctor wants to know more about the biopsy. Choosing this refinement option leads to a query graph Q_3 which is the upper query graph in Fig. 6. Only the attributes which are new in Q_3 are shown: we see that a biopsy of the lymph node revealed a melanoma in the groin, and that a dissection of the groin is scheduled. The doctor could maybe then go back to Q_2, navigating backwards through query history, and try a different refinement. There are two links "has finding" that could be connected to the X_{INV} vertex, connecting to CONDITION objects and RESULT objects, respectively. In [19], the RESULT type is described as "the numeric or qualitative finding of an INVESTIGATION, excluding CONDITION". The doctor wants to see what results, other than the melanoma, were obtained from investigations and now chooses the attribute "has_finding(X_{INV}) = X_{R}" from the list of refinement options. The resulting graph Q_4 is the lower graph in Fig. 6. It says that there was just one investigation producing a (negative) result, which was a PET scan.

5 Algorithm

The algorithm in Fig. 7 describes in more detail how a user query is processed by the system. The function cl takes a query graph (V, E, κ) and the associated LCF $((\mathbb{K}_t)_{t\in T}, L)$ as an input, and computes another query graph, the *closure* of (V, E, κ), along with a set $refs$ of refinement options. The *vlist* contains all vertices to be processed, initially it contains all vertices of the graph (line 1). The refinement set is initialized as the empty set (line 2).

Before the closure can be computed, we need the extent for each of the vertices (line 3). The algorithm is not explicit about how this is done, but a

```
      function cl((V, E, κ), ((𝕂_t)_{t∈T}, L)):
 1    vlist := V;
 2    refs := ∅;

 3    compute ext(x) for all x ∈ V;
 4    join equivalent vertices;

 5    while vlist ≠ ∅
 6      select x ∈ vlist;

 7      for each λ ∈ L with src(λ) = type(x)
 8        if (∄y ∈ V : (x, λ, y) ∈ E)
 9          if (∀g ∈ ext(x) : λ(g) exists)
10            select y ∈ Var_{dest(λ)} \ V;
11            V := V ∪ {y} ; E := E ∪ {(x, λ, y)};
12            ext(y) := λ(ext(x));
13            vlist := vlist ∪ {y};
14          else if (∃g ∈ ext(x) : λ(g) exists)
15            select y ∈ Var_{dest(λ)} \ V;
16            refs := refs ∪ {"λ(x) = y"};

17      for each λ ∈ L with dest(λ) = type(x)
18        if (∄y ∈ V : (y, λ, x) ∈ E)
19          if (∀g ∈ ext(x) : λ^{-1}(g) ≠ ∅)
20            select y ∈ Var_{src(λ)} \ V;
21            V := V ∪ {y} ; E := E ∪ {(y, λ, x)};
22            ext(y) := λ^{-1}(ext(x));
23            vlist := vlist ∪ {y};
24          else if (∃g ∈ ext(x) : λ^{-1}(g) ≠ ∅)
25            select y ∈ Var_{src(λ)} \ V;
26            refs := refs ∪ {"λ(y) = x"};

27      join equivalent vertices;
28      vlist := vlist \ {x};

29    for each x ∈ V
30      κ(x) := (ext(x)'', ext(x)');
31      for each many-valued attribute a of type(x)
32        for each C ⋖ κ_a(x)
33          if (ext(C) ≠ ∅)
34            refs := refs ∪ (int(C) \ int(κ_a(x)));
```

Fig. 7. Computing the closure of a query graph

straightforward approach is to process the query graph as an SQL query (see Sect. 6). The extent of each variable is the set of all distinct elements in the corresponding column of the result table.

If a query graph Q contains vertices x and y with $\text{type}(x) = \text{type}(y)$ and $\rho(x) = \rho(y)$ for all $\rho \in P(Q)$, these should be contracted to a single vertex,

and V and *vlist* modified accordingly (line 4, also line 27). Note that this is a stronger condition than $\mathrm{ext}(x) = \mathrm{ext}(y)$. The situation does not occur during the processing of Q_1, but the cycle in the graph Q_3 in Fig. 6 was obtained by such a contraction.

In the while loop (lines 5-28), actual or proposed attachments of vertices to the graph are computed. This is done separately for each $x \in vlist$. If $\lambda \in L$ is a link with $src(\lambda) = type(x)$, then an outgoing edge (x, λ, y) to a new vertex y is created (lines 10+11) if

1. the vertex x is not already connected to an outgoing λ-edge (line 8),
2. for all $g \in \mathrm{ext}(x)$ there exists an object h with $(g, h) \in \lambda$ (line 9).

At this stage we assume that $\kappa(y)$ is always the top concept for the respective type (no restrictions on the values of y). Then, by 2., attachment of y results in an extension of the previous graph, and the extent of y is given by $\mathrm{ext}(y) = \lambda(\mathrm{ext}(x))$. An assignment in line 12 is made accordingly. New vertices are inserted into *vlist* for subsequent processing (line 13). Note that this implies the possibility that the algorithm does not terminate; the output might converge against an infinite graph. The while loop should be aborted when the graph becomes too large.

If the above condition 2. does not hold, then no attachment to the graph is made. However, if the condition

3. for some but not all $g \in \mathrm{ext}(x)$ there exists an object h with $(g, h) \in \lambda$ (line 14),

holds, which also implies 1., then the corresponding attachment is stored as a refinement option (line 16).

The code on lines 17-26 repeats the previous steps for incoming edges: for each $\lambda \in L$ with $\mathrm{dest}(\lambda) = \mathrm{type}(x)$, a new vertex y is attached via an incoming edge (y, λ, x) if

1'. the vertex x is not already connected to an incoming λ-edge (line 18),
2'. for all $g \in \mathrm{ext}(x)$ there exists an object h with $(h, g) \in \lambda$ (line 19),

whereas a refinement option is generated if

3'. for some but not all $g \in \mathrm{ext}(x)$ there exists an object h with $(h, g) \in \lambda$ (line 24).

The inverse relation λ^{-1} is generally not a map, so the link λ may connect several objects h to the same object g. However, as condition 1'. states, at most one incoming λ-edge will be attached to x, and all objects h with $\lambda(h) \in \mathrm{ext}(g)$ are summarized by a single vertex y (cf. line 22). More specific queries involving several incoming λ-edges to x can be reached through user interaction, e.g. by duplicating y and then refining each copy of y individually. For example, the graph Q_5 in Fig. 8 is another extension of query Q_1; it is the most detailed extension. But as we have seen in the navigation example, the algorithm outputs Q_2, which is a high-level representation of Q_5. The three incoming edges on the

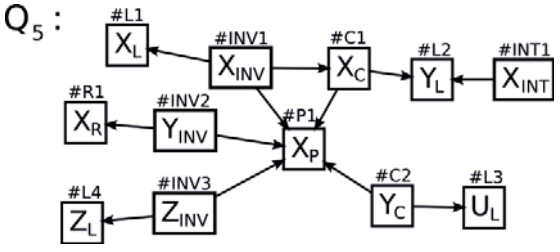


Fig. 8. Maximal extension (labels omitted)

left side of X_P in Q_5 are summarized by a single incoming edge in Q_2, and so are the two incoming edges and their respective attachments to the right of X_P.

The same considerations apply for outgoing edges if λ is a general relation and not a map. This allows to deal with set-valued attributes in the database, but alternatively a set-valued attribute can be eliminated by modeling it as a database table.

Note that conditions 1. and 1'. imply that the graph attachments computed by the algorithm can not be folded onto some other part of the graph. In particular, the closure of an unretractive graph is unretractive.

After the structure of the closure graph is established, the concept map κ is computed (lines `29+30`). In the output, we can write next to each vertex its intent $\mathrm{ext}(x)'$ as a label. In lines `31-34`, refinement terms are computed for each vertex. The refinement terms for a vertex x could be chosen to describe lower neighbor concepts of $\kappa(x)$. For example, if $\kappa(x)$ is the top concept of the INVESTIGATION lattice in Fig. 9 (left), the terms "$\mathrm{class}(x) \leq$ biopsy" and "$\mathrm{class}(x) \leq$ scan" describe the two lower neighbor concepts. The example highlights a problem: If the doctor was potentially interested in what investigations are scheduled, she would have no hint that there actually is a refinement term "$\mathrm{date}(x) =$ scheduled" deeper in the lattice.

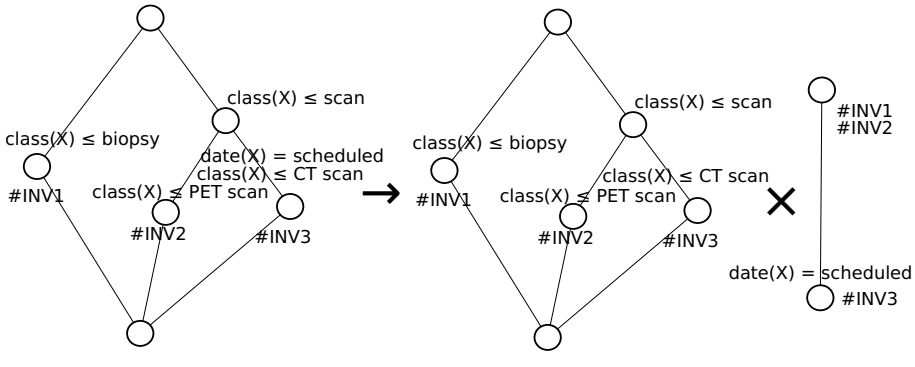


Fig. 9. Concept lattice for the INVESTIGATION type, decomposition

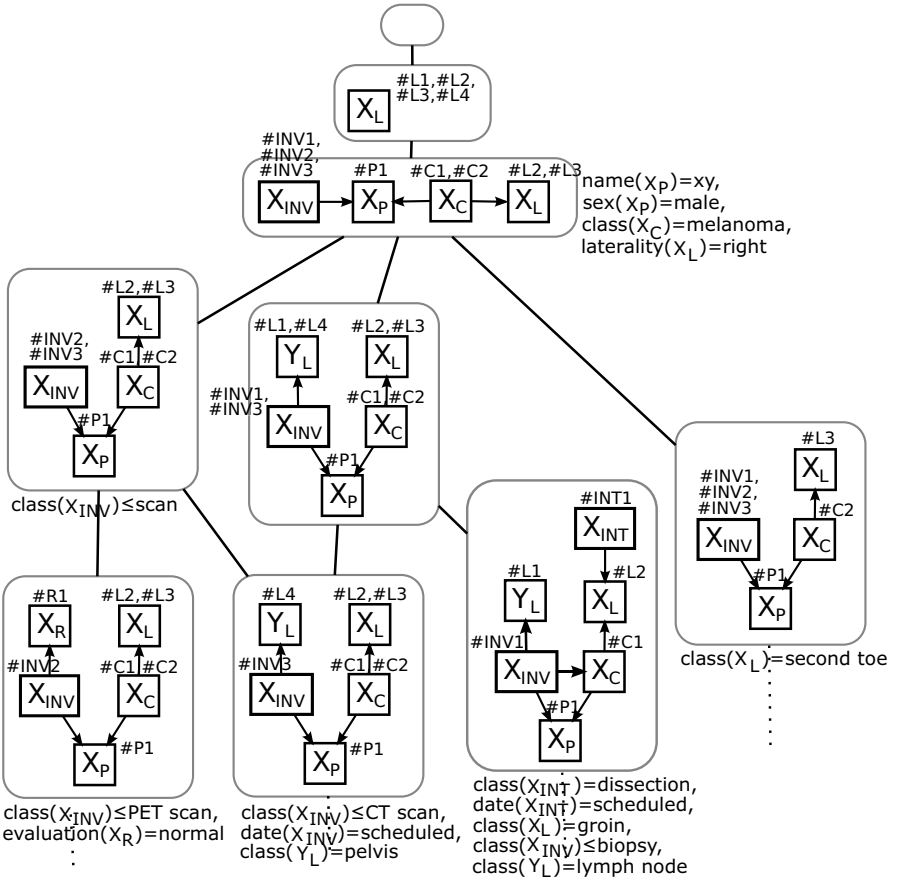


Fig. 10. Line diagram of all closures of unretractive query graphs (upper half only)

A way to address this problem in FCA-based navigation is to compute the refinement options separately for each many-valued attribute. In the INVESTIGATION example, "class" and "date" are the many-valued attributes. We define $\mathbb{K}_{\mathrm{type}(x),a}$ to be the context obtained from $\mathbb{K}_{\mathrm{type}(x)}$ by removing all columns unrelated to the many-valued attribute a. So $\mathbb{K}_{\mathrm{INV,class}}$ has four columns and $\mathbb{K}_{\mathrm{INV,date}}$ has one column (see Fig. 4). The corresponding lattices are shown in Fig. 9 (right). We further define $\kappa_a(x) := (ext(x)'', \mathrm{ext}(x)')$, where $'$ is the derivation operator of $\mathbb{K}_{\mathrm{type}(x),a}$. With this approach, which is used in lines `31-34`, a refinement option for x describes a lower neighbor of some $\kappa_a(x)$. This is how the first three refinement options listed under the X_P in Fig. 5 have been obtained.

Figure 10 has been included for reference. It shows the upper part of the partial order of all unretractive query graphs for the given LCF. The topmost element is supposed to be the empty graph. The bottom element is the graph from Fig. 8. The lower graphs, which are not shown here, result from combinations of branches attached to X_P in the graphs above.

6 Alternative Representations of Query Graphs

A query graph can be interpreted as a quantifier-free conjunctive formula in one or more variables. The conjunctive terms are statements about attribute values and follow a domain-specific syntax, e.g. the symbol "$\leq$" has been used to express a hierarchy of class names. If some of the variables were existence quantified, the algorithm in Sect. 5 could still be used for computing the closure (newly created vertices are then existence quantified by default), but only a subset of the refinement options would apply because a refinement must affect at least one free variable. The morphism definition would have to be adapted, too.

The following statement shows by example how a query graph (in this case Q_2) can be translated into SQL:

```
SELECT t1.id as X_INV, t2.id as X_P, t3.id as X_C, t4.id as X_L
FROM INVESTIGATION as t1, PATIENT as t2, CONDITION as t3,
        LOCATION as t4
WHERE t1.patient = t2.id AND t2.name = "xy" AND t2.sex = "male"
        AND t3.patient = t2.id AND t3.has_location = t4.id
        AND t3.class = "melanoma" AND t4.laterality = "right"
```

Table aliases are optional in this case, but they have to be used if two or more vertices are associated with the same table. The example lacks full generality because the exact form of `WHERE`-clauses depends on domain-specific syntax. A conjunctive clause like "class$(x) \leq$ scan" can be easily translated into a `WHERE`-clause if the database contains a nested set representation of the hierarchy [4].

Query graphs can also be considered a subset of the graph-based query language SPARQL. A query graph corresponds to a collection of triples of URIs and literals. The expression of class hierarchies is natively supported by the `rdfs:subClassOf` property. Concurrent research in this area will have to be compared with the approach presented in this paper.

7 Related Work

In [15], an extension of Formal Concept Analysis is presented which bases on conditions almost identical to those considered in this article. A Relational Context Family is introduced as a pair $(\mathbf{K}_R, A_R)$ consisting of a set $\mathbf{K}_R$ of many-valued contexts and a set A_R of many-to-many relations, modeled as set-valued functions, between the object sets of contexts in $\mathbf{K}_R$. The contexts in an LCF $((\mathbb{K}_t)_{t\in T}, L)$ are binary, but this is not a significant difference because they are also derived from many-valued contexts. Another difference between RCF and LCF is that the elements of L model many-to-*one* relations. A many-to-many relation can be decomposed into a one-to-many and a many-to-one relation by introducing relation instances as an intermediary object type, and the definition of the LCF was motivated by the idea that such a decomposition is also

semantically meaningful. This is the reason why the elements of L are refered to as "links" rather than relations.

An algorithm is presented in [15] which transforms the RCF into a set of lattices, one for each many-valued context, where relations between objects are considered for the definition of concepts. The extents of the concepts defined by Huchard et al. w.r.t. one of their methods of relational scaling are precisely the extents $\mathrm{ext}(x)$ of variables x within tree-structured query graphs. This can be shown by induction over the depth of trees with root x and edges in either direction.

Query graphs do also resemble the abstract concept graphs which have been defined by Wille in [21], but most later definitions of concept graphs incorporate a particular realization in their definition, which leads to a different kind of order on the graphs that is based on "conceptual content" [22]. In [8], concept graphs have been used for querying flight information.

In [6], SQL queries which may contain negation and subqueries are visually represented by concept graphs. These graphs, which are called nested concept graphs with cuts, have considerably greater expressive power than the query graphs which have been presented here. They have not been considered for browsing, however.

In [10], a browsing application is presented which seems to rely on the same extension mechanism to introduce linked objects. A SPARQL-like syntax is used which also allows to express negation and disjunction. However, querying is menu-based and centers on a single object, whereas the application proposed here uses a purely graph-based interface and has a theoretical underpinning based on graph homomorphism which leads to a preorder on query graphs.

The (pre-)order on Basic Conceptual Graphs described in [5] is essentially the same that is used for query graphs. The book also features generalization and specialization operations with respect to this order.

8 Conclusion

An idea for exploring the content of a relational database based on an extension of FCA which considers links between objects of different types has been presented. The advantage of this approach is that the conceptual structure defined over an LCF reflects the schema of the database to a high degree, which facilitates the integration of data modeling efforts into the general system. An algorithm has been given which explains how a query graph can be refined to a more informative, summarized view of the underlying data without exposing too much information at once.

There are several mathematical questions which still need to be answered, e.g. if the algorithm for query graph extension could possibly lead to infinite graphs by means of structural self-replication. A prototype in Python is currently under development which will implement the idea presented here and will allow a better evaluation.

References

1. Acciarri, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In: Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference, pp. 1670–1671. AAAI Press / The MIT Press (2005)
2. Becker, P., Hereth Correia, J.: Software tools for formal concept analysis. In: Hitzler, P., Schärfe, H. (eds.) Conceptual Structures in Practice, pp. 47–72. Chapman & Hall/CRC (2009)
3. Calvanese, D., Keet, C.M., Nutt, W., Rodriguez-Muro, M., Stefanoni, G.: Web-based graphical querying of databases through an ontology: The WONDER system. In: SAC 2010: Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1388–1395. ACM, New York (2010)
4. Celko, J.: SQL for Smarties, 2nd edn. Morgan Kaufmann, San Francisco (2000)
5. Chein, M., Mugnier, M.L.: Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Advanced Information and Knowledge Processing. Springer, London (2009)
6. Hereth Correia, J., Dau, F.: Nested concept graphs: Applications for databases and mathematical foundations. In: de Moor, A., Ganter, B. (eds.) Using Conceptual Structures: Contributions to ICCS 2003, pp. 133–150. Shaker Verlag, Aachen (2003)
7. Ducrou, J., Eklund, P., Wilson, T.: An intelligent user interface for browsing and searching MPEG-7 images using concept lattices. In: Ben Yahia, S., Mephu Nguifo, E., Belohlavek, R. (eds.) CLA 2006. LNCS (LNAI), vol. 4923, pp. 1–21. Springer, Heidelberg (2008)
8. Eklund, P.W., Groh, B., Stumme, G., Wille, R.: A contextual-logic extension of TOSCANA. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867, pp. 453–467. Springer, Heidelberg (2000)
9. Ferré, S.: CAMELIS: Organizing and browsing a personal photo collection with a logical information system. In: Diatta, J., Eklund, P., Liquière, M. (eds.) Proceedings of CLA 2007. CEUR Workshop Proceedings, vol. 331, pp. 112–123 (2007)
10. Ferré, S.: Conceptual navigation in RDF graphs with SPARQL-like queries. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA 2010. LNCS, vol. 5986, pp. 193–208. Springer, Heidelberg (2010)
11. Ferré, S., Ridoux, O., Sigonneau, B.: Arbitrary relations in formal concept analysis and logical information systems. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) ICCS 2005. LNCS (LNAI), vol. 3596, pp. 166–180. Springer, Heidelberg (2005)
12. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer, Berlin (1999)
13. Godin, R., Gecsei, J., Pichet, C.: Design of a browsing interface for information retrieval. In: Belkin, N.J., van Rijsbergen, C.J. (eds.) Proceedings of SIGIR 1989, 12th International Conference on Research and Development in Information Retrieval, pp. 32–39. ACM Press, New York (1989)
14. Hereth, J.: Relational scaling and databases. In: Priss, U., Corbett, D.R., Angelova, G. (eds.) ICCS 2002. LNCS (LNAI), vol. 2393, pp. 62–76. Springer, Heidelberg (2002)
15. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Annals of Mathematics and Artificial Intelligence 49(1-4), 39–76 (2007)
16. Prediger, S.: Logical scaling in formal concept analysis. In: Lukose, D., Delugach, H.S., Keeler, M., Searle, L., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 332–341. Springer, Heidelberg (1997)

17. Prediger, S., Stumme, G.: Theory-driven logical scaling. In: Proc. 6th Intl. Workshop Knowledge Representation Meets Databases, Heidelberg. CEUR Workshop Proc., pp. 46–49 (1999)
18. Priss, U.: Establishing connections between formal concept analysis and relational databases. In: Dau. F., Mugnier, M.-L., Stumme, G. (eds.) Common Semantics for Sharing Knowledge: Contributions to ICCS 2005, pp. 132–145 (2005)
19. Roberts, A., Gaizauskas, R., Hepple, M., Demetriou, G., Guo, Y., Setzer, A., Roberts, I.: Semantic annotation of clinical text: The CLEF corpus. In: Proceedings of the LREC Workshop on Building and Evaluating Resources for Biomedical Text Mining, pp. 19–26 (2008)
20. Stumme, G., Wille, R., Wille, U.: Conceptual knowledge discovery in databases using formal concept analysis methods. In: Zytkow, J.M., Quafafou, M. (eds.) PKDD 1998. LNCS, vol. 1510, pp. 450–458. Springer, Heidelberg (1998)
21. Wille, R.: Conceptual graphs and formal concept analysis. In: Lukose, D., Delugach, H.S., Keeler, M., Searle, L., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 290–303. Springer, Heidelberg (1997)
22. Wille, R.: Formal concept analysis and contextual logic. In: Hitzler, P., Schärfe, H. (eds.) Conceptual Structures in Practice, pp. 137–173. Chapman & Hall/CRC (2009)

On Factorization of Concept Lattices by Incompatible Tolerances*

Michal Krupka

Department of Computer Science
Palacký University in Olomouc
17. listopadu 12, CZ-77146 Olomouc
Czech Republic
michal.krupka@upol.cz

Abstract. It is a well-known fact that complete tolerance relations on concept lattices are in one-to-one correspondence with some superrelations (called block relations) of the incidence relation of the underlying formal context. However, sometimes it is useful to consider more general superrelations of the incidence relation, leading to tolerance relations, not compatible with the lattice structure of the concept lattice. In this paper, we give a characterization of such tolerances and present a mathematical framework for factorizing any complete lattice by such incompatible tolerances.

Keywords: Concept lattice, Tolerance, Completion, Factorization.

1 Introduction

In this paper, we study the possibilities of reducing the size of concept lattices by means of factorization. It is a well-known fact [6] that for a formal context $\langle X, Y, I \rangle$ and its concept lattice $\mathscr{B}(X, Y, I)$, complete tolerance relations on $\mathscr{B}(X, Y, I)$ are in one-to-one correspondence with so-called *block relations* $J \supseteq I$. As it is known from the theory of lattices [5,6], factorization of complete lattices by complete tolerance relations has the advantage that it preserves the completeness of the lattice. Moreover, in the case of concept lattices, the factor lattice of $\mathscr{B}(X, Y, I)$ by the complete tolerance relation induced by a block relation $J \supseteq I$ is isomorphic to the concept lattice $\mathscr{B}(X, Y, J)$. These results allow us to reduce the size of concept lattices by putting together formal concepts, which are (as maximal rectangles in the data table) contained within a formal concept of some augmented incidence relation.

However, the condition on a superrelation $J \supseteq I$ to be a block relation is sometimes too limiting. For example, in [4] an interesting and natural method is introduced to reduce the size of a concept lattice by using an equivalence relation on the set of attributes. However, such an equivalence induces an equivalence on the concept lattice, which is not a congruence.

Main results of this paper are the following. We formulate the problem of the minimal completion of an ordered set endowed with a tolerance relation and solve this

* Supported by grant no. P103/10/1056 of the Czech Science Foundation.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 167–182, 2011.

problem by identifying ordered sets with tolerance with appropriate fuzzy ordered sets and then performing the well-known Dedekind-MacNeille completion of fuzzy ordered sets. Then we characterize all tolerance relations on concept lattices which are induced by superrelations of the incidence relation and show that factorization of concept lattices by such tolerances consists of two steps: a completion of the concept lattice with the tolerance and factorization of the resulting lattice by the completion of the tolerance (which is now a complete tolerance).

Due to limited space, we abbreviate, resp. omit, some proofs.

2 Preliminaries from Lattice Theory and Formal Concept Analysis

In this section, we briefly recall basic facts on factorization of complete lattices and Formal Concept Analysis (FCA). Details can be found in [6].

Recall that a *tolerance on a set* X is a relation $\sim$ which is reflexive and symmetric. Each tolerance induces a covering of X, called the *factor* (*quotient*) *set*. This covering consists of all maximal blocks of the tolerance, i.e., maximal (with respect to set inclusion) subsets $B \subseteq X$ such that for any $a, b \in B$ it holds $a \sim b$. The factor set of X induced by a tolerance $\sim$ is denoted $X/\sim$. Note that $X/\sim$ is a covering of X, but need not be a partition. $X/\sim$ is a partition of X if and only if $\sim$ is transitive (hence an equivalence relation).

A *complete tolerance relation* on a complete lattice $\mathbf{L} = \langle L, \wedge, \vee, 0, 1\rangle$ is a tolerance which preserves suprema and infima. More precisely, a tolerance $\sim$ on $\mathbf{L}$ is complete if from $a_j \sim b_j$ for all $j \in J$ it follows $\bigvee_{j\in J} a_j \sim \bigvee_{j\in J} b_j$ and $\bigwedge_{j\in J} a_j \sim \bigwedge_{j\in J} b_j$ (J is an arbitrary index set).

For a complete tolerance $\sim$ on $\mathbf{L}$ and $a \in L$ we denote

$$a^{\sim} = \bigvee\{b \in L \mid a \sim b\}, \quad a_{\sim} = \bigwedge\{b \in L \mid a \sim b\}, \tag{1}$$

$$[a]_{\sim} = [a_{\sim}, (a_{\sim})^{\sim}], \quad [a]^{\sim} = [(a^{\sim})_{\sim}, a^{\sim}] \tag{2}$$

($[a_1, a_2]$ denotes the interval $\{b \in L \mid a_1 \leq b \leq a_2\}$).

The equations (2) describe all maximal blocks of $\sim$ [5, 12]: it holds $L/\sim = \{[a]_{\sim} \mid a \in L\} = \{[a]^{\sim} \mid a \in L\}$.

An ordering on the set $L/\sim$ is introduced using suprema of maximal blocks and can be equivalently introduced using infima. For blocks $B_1, B_2 \in L/\sim$ we set

$$B_1 \leq B_2 \quad \text{iff} \quad \bigvee B_1 \leq \bigvee B_2 \quad \left(\text{iff} \quad \bigwedge B_1 \leq \bigwedge B_2\right). \tag{3}$$

The set $L/\sim$ together with this ordering is a complete lattice, which is denoted by $\mathbf{L}/\sim$.

Formal Concept Analysis has been introduced in [11], the basic reference is [6]. A *formal context* is a triple $\langle X, Y, I\rangle$ where X is a set of objects, Y a set of attributes and $I \subseteq X \times Y$ a binary relation between X and Y. For $\langle x, y\rangle \in I$ it is said "The object x has the attribute y".

For subsets $A \subseteq X$ and $B \subseteq Y$ we set

$$A^{\uparrow I} = \{y \in Y \mid \text{for each } x \in A \text{ it holds } \langle x,y\rangle \in I\},$$
$$B^{\downarrow I} = \{x \in X \mid \text{for each } y \in B \text{ it holds } \langle x,y\rangle \in I\}.$$

If $A^{\uparrow I} = B$ and $B^{\downarrow I} = A$, then the pair $\langle A,B\rangle$ is called a *formal concept* of $\langle X,Y,I\rangle$. The set A is called the *extent of* $\langle A,B\rangle$, the set B the *intent of* $\langle A,B\rangle$.

We write $^{\uparrow}$ (resp. $^{\downarrow}$) instead of $^{\uparrow I}$ (resp. $^{\downarrow I}$) when I is obvious.

A partial order $\leq$ on the set $\mathscr{B}(X,Y,I)$ of all formal concepts of $\langle X,Y,I\rangle$ is defined by

$$\langle A_1,B_1\rangle \leq \langle A_2,B_2\rangle \quad \text{iff} \quad A_1 \subseteq A_2 \quad (\text{iff } B_2 \subseteq B_1).$$

$\mathscr{B}(X,Y,I)$ together with $\leq$ is a complete lattice and is called the *concept lattice* of $\langle X,Y,I\rangle$.

3 Fuzzy Logic and Fuzzy Ordered Sets

In this section, we introduce briefly basic notions of fuzzy logic and fuzzy ordered sets as needed in the paper. The reader can refer to [3] for details. Note that we use these notions (and Formal Concept Analysis in fuzzy setting, introduced in the next section) merely as a tool; all the main results of this paper belong to classical (non-fuzzy) logic and classical FCA.

As the structure of truth degrees, we use complete residuated lattices. A *complete residuated lattice* is an algebra $\mathbf{L} = \langle L,\wedge,\vee,\otimes,\rightarrow,0,1\rangle$, where $\langle L,\wedge,\vee,0,1\rangle$ is a complete lattice with the least element 0 and the greatest element 1; $\langle L,\otimes,1\rangle$ is a commutative monoid (i.e., $\otimes$ is commutative, associative, and $a\otimes 1 = 1\otimes a = a$ for each $a \in L$); $\otimes$ *(product)* and $\rightarrow$ *(residuum)* satisfy the so-called *adjointness property*: $a\otimes b \leq c$ iff $a \leq b \rightarrow c$ for each $a,b,c \in L$ (the order $\leq$ on L is defined as usual by $a \leq b$ iff $a\wedge b = a$). Elements of L are called *truth degrees*. $\otimes$ and $\rightarrow$ are truth functions (interpretations) of "fuzzy conjunction" and "fuzzy implication".

The biresiduum is a binary operation $\leftrightarrow$ on L defined by $a \leftrightarrow b = (a \rightarrow b)\wedge(b \rightarrow a)$.

For an integer $n > 0$, the *n-th power* a^n *of* $a \in L$ is defined by $a^1 = a$, $a^{n+1} = a^n \otimes a$. We also set for $a > 0$, $a^0 = 1$ and for $a < 1$, $a^\infty = 0$.

A common choice of $\mathbf{L}$ is a structure with $L = [0,1]$ (unit interval), $\wedge$ and $\vee$ being minimum and maximum, $\otimes$ being a left-continuous t-norm with the corresponding $\rightarrow$.

Three most important pairs of adjoint operations on the unit interval are:

$$\text{Łukasiewicz:} \quad \begin{aligned} a\otimes b &= \max(a+b-1,0),\\ a\rightarrow b &= \min(1-a+b,1),\end{aligned} \tag{4}$$

$$\text{Gödel:} \quad \begin{aligned} a\otimes b &= \min(a,b),\\ a\rightarrow b &= \begin{cases} 1 & \text{if } a \leq b,\\ b & \text{otherwise,}\end{cases}\end{aligned} \tag{5}$$

$$\text{Goguen:} \quad \begin{aligned} a\otimes b &= a\cdot b,\\ a\rightarrow b &= \begin{cases} 1 & \text{if } a \leq b,\\ \frac{b}{a} & \text{otherwise.}\end{cases}\end{aligned} \tag{6}$$

Complete residuated lattices on $[0,1]$ given by (4), (5), and (6) are called *the standard Łukasiewicz, Gödel, Goguen (product) algebras*, respectively. A special case of a complete residuated lattice is also the *two-element Boolean algebra* $\mathbf{2} = \langle \{0,1\}, \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$, which is the structure of truth degrees of classical logic.

In this paper, we use basic properties of residuated lattices as summarized, for example, in [3].

Any element $e \in L$ defines a complete tolerance relation $\sim_e$ on L by

$$a \sim_e b \quad \text{iff} \quad e \leq a \leftrightarrow b.$$

In [9] a structure of complete residuated lattice on the factor set L/e is introduced by setting for any $B_1, B_2 \in L/e$,

$$B_1 \otimes B_2 = \left[\bigvee B_1 \otimes \bigvee B_2 \right]_{\sim_e}, \tag{7}$$

$$B_1 \rightarrow B_2 = \left[\bigvee B_1 \rightarrow \bigvee B_2 \right]_{\sim_e}. \tag{8}$$

Now the set L/e, together with elements $0, 1 \in L/e$ and operations $\wedge, \vee$ given by the factor lattice structure, and together with operations $\otimes, \rightarrow$, introduced in (7) and (8), is a complete residuated lattice, which is denoted by $\mathbf{L}/e$.

An $\mathbf{L}$-*set* (a *fuzzy set*) A *in universe* X is a mapping $A : X \rightarrow L$. Values $A(x) \in L$ are interpreted as "the degree to which x belongs to A". The set of all $\mathbf{L}$-sets in universe X is denoted by L^X.

For $\mathbf{L}$-sets $A, B \in L^X$, put

$$S(A,B) = \bigwedge_{x \in X} A(x) \rightarrow B(x), \tag{9}$$

$$A \approx^X B = \bigwedge_{x \in X} A(x) \leftrightarrow B(x). \tag{10}$$

$S(A,B)$ and $A \approx^X B$ are called *the degree of subsethood of* A *in* B and *the degree of equality of* A *and* B, respectively.

For $\mathbf{L} = \mathbf{2}$ (the two-element Boolean algebra), $\mathbf{L}$-sets can be identified with classical sets by identifying sets with their characteristic functions.

A *binary* $\mathbf{L}$-*relation between sets* X, Y is an $\mathbf{L}$-set $R \in L^{X \times Y}$, where $R(x,y)$ is interpreted as the degree to which x and y are in R. $\mathbf{L}$-relations are often written using infix notation as in $xRy \in L$. A binary $\mathbf{L}$-relation $R : X \times X \rightarrow L$ is called a binary $\mathbf{L}$-relation *on the set* X.

A binary $\mathbf{L}$-relation R on a set X is called *reflexive* if $R(x,x) = 1$ for any $x \in X$, *symmetric* if $R(x,y) = R(y,x)$ for any $x, y \in X$, *crisply antisymmetric* if from $R(x,y) = R(y,x) = 1$ it follows $x = y$ for any $x, y \in X$, and *transitive* if $R(x,y) \otimes R(y,z) \leq R(x,z)$ for any $x, y, z \in X$. R is called an $\mathbf{L}$-*equivalence* if it is reflexive, symmetric and transitive. If, moreover, for any $x, y \in X$ from $R(x,y) = 1$ it follows $x = y$, then R is called an $\mathbf{L}$-*equality* on X. $\mathbf{L}$-equalities are often denoted by $\approx$. The similarity $\approx^X$ of $\mathbf{L}$-sets, introduced before, is an $\mathbf{L}$-equality on L^X.

Let $\sim$ be an $\mathbf{L}$-equivalence on X, R an $\mathbf{L}$-relation on X. We say that R is *compatible with* $\sim$ if the following condition holds for any $x, x', y, y' \in X$:

$$R(x,y) \otimes (x \sim x') \otimes (y \sim y') \leq R(x', y').$$

For an **L**-set $A \in L^X$ and $a \in L$, the *a-cut of A* is a crisp subset ${}^aA \subseteq X$ such that $x \in {}^aA$ iff $a \leq A(x)$. This definition applies also to binary **L**-relations, whose a-cuts are classical (crisp) binary relations.

An **L**-*order* on a set V with **L**-equality relation $\approx$ [1,3] is a binary **L**-relation $\preceq$ which is compatible with $\approx$, reflexive, transitive and satisfies $(v \preceq w) \wedge (w \preceq v) \leq v \approx w$ for any $v, w \in V$ (*antisymmetry*). If $\preceq$ is an **L**-order on V with **L**-equality $\approx$, then the tuple $\mathbf{V} = \langle\langle V, \approx\rangle, \preceq\rangle$ is called an **L**-*ordered set*. An immediate consequence of the definition is that for any $v, w \in V$ it holds

$$v \approx w = (v \preceq w) \wedge (w \preceq v). \tag{11}$$

For any reflexive, crisply antisymmetric, and transitive **L**-relation $\preceq$ on a set V, (11) defines an **L**-equality $\approx$ on V such that $\langle\langle V, \approx\rangle, \preceq\rangle$ is an **L**-ordered set.

For two **L**-ordered sets $\mathbf{V} = \langle\langle V, \approx_V\rangle, \preceq_V\rangle$ and $\mathbf{W} = \langle\langle W, \approx_W\rangle, \preceq_W\rangle$, a mapping $F : V \to W$ is called an *embedding of* $\mathbf{V}$ *into* $\mathbf{W}$, if it satisfies

$$(v_1 \preceq v_2) = (F(v_1) \preceq F(v_2)),$$

for any $v_1, v_2 \in V$. F is an *isomorphism from* $\mathbf{V}$ *to* $\mathbf{W}$, if it is bijective and both F and F^{-1} are embeddings. $\mathbf{V}$ and $\mathbf{W}$ are called *isomorphic*, if there exists an isomorphism from $\mathbf{V}$ to $\mathbf{W}$.

For an **L**-ordered set $\mathbf{V} = \langle\langle V, \approx\rangle, \preceq\rangle$ and an **L**-set $U \in L^V$ we define **L**-sets $\mathscr{U}U$, $\mathscr{L}U \in L^V$ by

$$\mathscr{L}U(v) = \bigwedge_{w \in V} U(w) \to (v \preceq w),$$
$$\mathscr{U}U(v) = \bigwedge_{w \in V} U(w) \to (w \preceq v).$$

$\mathscr{L}U$ (resp. $\mathscr{U}U$) is called *the lower cone* (resp. *the upper cone*) of U.

For any **L**-set $W \in L^V$ there exists at most one element $v \in V$ such that $\mathscr{L}W(v) \wedge \mathscr{U}(\mathscr{L}W)(v) = 1$ (resp. $\mathscr{U}W \wedge \mathscr{L}(\mathscr{U}W)(v) = 1$) [1,3]. If there is such an element, we call it *the infimum of* W (resp. *the supremum of* W) and denote $\inf W$ (resp. $\sup W$); otherwise we say, that the infimum (resp. supremum) does not exist. An **L**-ordered set $\mathbf{V} = \langle\langle V, \approx\rangle, \preceq\rangle$ is called *completely lattice* **L**-*ordered*, if for any $U \in L^V$, both $\inf U$ and $\sup U$ exist.

For a completely lattice **L**-ordered set $\mathbf{V} = \langle\langle V, \approx\rangle, \preceq\rangle$, an **L**-set $U \in L^V$ is said to be *infimum-dense*, if for any $v \in V$ there exists an **L**-set $W \in L^V$ such that $W \subseteq U$ and $v = \inf W$. Similarly, U is called *supremum-dense*, if for any $v \in V$ there exists an **L**-set $W \in L^V$ such that $W \subseteq U$ and $v = \sup W$.

Lower and upper cones can be used for constructing the so called Dedekind-MacNeille completion of an **L**-ordered set [1,3]. The construction is a generalization of the well-known construction from ordered set theory and goes as follows. For an **L**-ordered set $\mathbf{U} = \langle\langle U, \approx_{\mathbf{U}}\rangle, \preceq_{\mathbf{U}}\rangle$ we set $\mathbf{V} = \langle\langle V, \approx_{\mathbf{V}}\rangle, \preceq_{\mathbf{V}}\rangle$, where $V = \{\langle A, B\rangle \in L^U \times L^U \mid \mathscr{U}A = B \text{ and } \mathscr{L}B = A\}$, $\langle A_1, B_1\rangle \approx_{\mathbf{V}} \langle A_2, B_2\rangle = A_1 \approx^U A_2$, $\langle A_1, B_1\rangle \preceq_{\mathbf{V}} \langle A_2, B_2\rangle = S(A_1, A_2)$. Further we define a mapping $f : U \to V$ by setting $f(u) = \langle \mathscr{L}\{u\}, \mathscr{U}\{u\}\rangle$.

The main result is that $\mathbf{V}$ is a completely lattice **L**-ordered set, f is an embedding of **L**-ordered sets and for any other completely lattice **L**-ordered set $\mathbf{V}'$ and embedding $f' : \mathbf{U} \to \mathbf{V}'$ there exists an embedding $h : \mathbf{V} \to \mathbf{V}'$ such that $f' = h \circ f$.

4 Formal Concept Analysis in Fuzzy Setting

In this section, we introduce the basic facts from Formal Concept Analysis in fuzzy setting. The reader can refer to [3] for details. Some more recent results can be found in [8,9].

By a *formal* **L**-*context* we understand the triple $\langle X, Y, I \rangle$, where X and Y are sets and I is an **L**-relation between X and Y, $I : X \times Y \to L$. The sets X and Y are interpreted as sets of objects, resp. attributes, and for any $x \in X$, $y \in Y$ the value $I(x,y) \in L$ is interpreted as the degree to which the object x has the attribute y.

For any **L**-set $A \in L^X$ of objects we define an **L**-set $A^\uparrow \in L^Y$ of attributes by

$$A^\uparrow(y) = \bigwedge_{x \in X} A(x) \to I(x,y). \tag{12}$$

Similarly, for any **L**-set $B \in L^Y$ of attributes we define an **L**-set $B^\downarrow$ of objects by

$$B^\downarrow(x) = \bigwedge_{y \in Y} B(y) \to I(x,y). \tag{13}$$

The **L**-set $A^\uparrow$ is interpreted as the **L**-set of all attributes shared by objects from A. Similarly, the **L**-set $B^\downarrow$ is interpreted as the **L**-set of all objects having the attributes from B in common.

An **L**-*formal concept* of a formal **L**-context $\langle X, Y, I \rangle$ is a pair $\langle A, B \rangle \in L^X \times L^Y$ such that $A^\uparrow = B$ and $B^\downarrow = A$. A is called *the extent*, B *the intent* of $\langle A, B \rangle$. The set of all formal concepts of $\langle X, Y, I \rangle$ is denoted $\mathscr{B}(X, Y, I)$ and called the **L**-*concept lattice of* $\langle X, Y, I \rangle$.

For any two formal concepts $\langle A_1, B_1 \rangle, \langle A_2, B_2 \rangle \in \mathscr{B}(X, Y, I)$ we set

$$(\langle A_1, B_1 \rangle \approx \langle A_2, B_2 \rangle) = (A_1 \approx^X A_2), \tag{14}$$

$$(\langle A_1, B_1 \rangle \preceq \langle A_2, B_2 \rangle) = S(A_1, A_2). \tag{15}$$

$\approx$ is an **L**-equality on $\mathscr{B}(X, Y, I)$ and $\preceq$ is an **L**-order on $\langle \mathscr{B}(X, Y, I), \approx \rangle$.

Note that the notions of (classical) formal context, formal concept and concept lattice represent special cases of the notions introduced here for **L** equal to the two-element Boolean algebra **2**. To distinguish between classical and fuzzy notions, we always use the prefix "**L**-" for fuzzy formal contexts, formal concepts and concept lattices.

The following theorem represents a basic result of Formal Concept Analysis in fuzzy setting. It first appeared in [1], our version is from [8].

Theorem 1 (Main theorem of fuzzy concept lattices). *1.* $\mathscr{B}(X, Y, I)$ *together with the* **L**-*order* $\preceq$ *is a completely lattice* **L**-*ordered set.*

2. A completely lattice **L**-*ordered set* $\mathbf{V} = \langle \langle V, \approx \rangle, \preceq \rangle$ *is isomorphic to an* **L**-*concept lattice* $\mathscr{B}(X, Y, I)$*, if and only if there are mappings* $\gamma : X \to V$ *and* $\mu : Y \to V$ *such that* $\gamma(X)$ *is supremum-dense in* $\mathbf{V}$*,* $\mu(Y)$ *is infimum-dense in* $\mathbf{V}$ *and for any* $x \in X$*,* $y \in Y$ *it holds*

$$I(x,y) = \gamma(x) \preceq \mu(y). \tag{16}$$

In [2], a method of reducing the size of fuzzy concept lattices by factorization is introduced. For any user-chosen threshold $e \in L$, the e-cut ${}^{e}\approx$ of the $\mathbf{L}$-equality $\approx$ on $\mathscr{B}(X,Y,I)$ is a complete tolerance. Thus, it is possible to use the method introduced in [5, 12] and construct the factor lattice $\mathscr{B}(X,Y,I)/{}^{e}\approx$, which is also denoted simply by $\mathscr{B}(X,Y,I)/e$. The following result has been proved in [9]:

Theorem 2. *$\mathscr{B}(X,Y,I)/e$ is isomorphic to $\mathscr{B}(X,Y,[I]^e)$ where the $\mathbf{L}$-relation $[I]^e : X \times Y \to L/e$ is defined by $[I]^e(x,y) = [I(x,y)]^{\sim_e}$.*

5 Posets with Tolerance

A *partially ordered set with tolerance* (or, simply, a *poset with tolerance*) is a structure $\mathbf{U} = \langle U, \leq_{\mathbf{U}}, \sim_{\mathbf{U}} \rangle$ where $\langle U, \leq_{\mathbf{U}} \rangle$ is a partially ordered set and $\sim_{\mathbf{U}}$ is a tolerance on U. $\mathbf{U}$ is called a *poset with equivalence*, if $\sim_{\mathbf{U}}$ is an equivalence.

We usually omit the subscript in $\leq_{\mathbf{U}}$ and $\sim_{\mathbf{U}}$ and denote the order resp. tolerance of $\mathbf{U}$ by $\leq$ resp. $\sim$.

Posets with tolerance can be depicted by Hasse diagrams with dashed ovals, denoting maximal blocks of the tolerance $\sim$ (see some figures below).

When working with the notions introduced below, we sometimes consider the set of all nonnegative integers together with a unique value ∞, where we set $n < \infty$ for any integer n.

For any elements u_1, u_2 of a poset with tolerance $\mathbf{U}$ we set $u_1 \leq^0_{\mathbf{U}} u_2$ iff $u_1 \leq u_2$ and for any integer $n > 0$ we set $u_1 \leq^n_{\mathbf{U}} u_2$ iff there exist elements $v_1, v_2 \in U$ such that $u_1 \leq^{n-1}_{\mathbf{U}} v_1$, $v_1 \sim v_2$, and $v_2 \leq u_2$. Finally we set $u_1 \leq^\infty_{\mathbf{U}} u_2$ for any $u_1, u_2 \in U$.

As an example, in Fig. 1 a situation where $u_1 \leq^2_{\mathbf{U}} u_2$ is depicted.

We say that the tolerance $\sim$ *satisfies the diagonal property*, if the following holds for any $u_1, u_2, u_3, u_4, v_1, v_2 \in U$:

$$\text{If } u_1 \geq v_1 \geq u_3,\ u_2 \geq v_2 \geq u_4,\ u_1 \sim u_4, \text{ and } u_2 \sim u_3, \text{ then } v_1 \sim v_2.$$

If for a poset with tolerance $\mathbf{U} = \langle U, \leq, \sim \rangle$ the tolerance $\sim$ satisfies the diagonal property, then we also say that $\mathbf{U}$ satisfies this property.

The diagonal property resembles the well-known situation from basic geometry where all segments inside any rectangle are at most as long as the diagonals (see Fig. 2). Hence the name "diagonal property".

The diagonal property can be equivalently formulated using the above relation $\leq^1_\sim$ as follows. For any $v_1, v_2 \in U$ from $v_1 \leq^1_\sim v_2$ and $v_2 \leq^1_\sim v_1$ it follows $v_1 \sim v_2$.

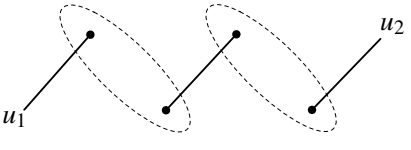


Fig. 1. $u_1 \leq^2_{\mathbf{U}} u_2$

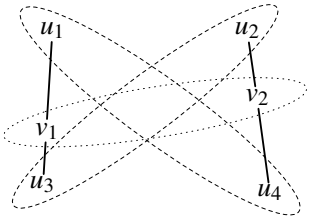


Fig. 2. The diagonal property: if elements in dashed ovals are related, then those in the dotted one are also related

We say that the tolerance $\sim$ *satisfies the strong diagonal property*, if for any integer $n > 0$ from $v_1 \leq^n_\sim v_2$ and $v_2 \leq^n_\sim v_1$ it follows $v_1 \sim v_2$.

For $k \in \{0,1,2,\ldots\} \cup \{\infty\}$, a poset with tolerance $\mathbf{U} = \langle U, \leq, \sim \rangle$ is said to be *k-complete*, if it satisfies the following conditions:

1. $\langle U, \leq \rangle$ is a complete lattice,
2. $\sim$ is a complete tolerance on $\langle U, \leq \rangle$,
3. $\sim^k = U \times U$.

If $k = \infty$, then Condition 3 is trivially satisfied. In this case, $\mathbf{U}$ is also called simply *complete*. Note that the motivation for the case $k < \infty$ is that it allows us to formulate and to prove main results of this paper. In the light of these results, this case seems to be quite appropriate.

Condition 3 can be equivalently reformulated as follows.

3a. If u_1 (resp. u_0) is the greatest (resp. the least) element of U, then $u_1 \sim^k u_0$.

For $k \in \{1,2,\ldots\} \cup \{\infty\}$, a mapping $f : \mathbf{U} \to \mathbf{V}$ of two posets with tolerance is called a *k-embedding*, if for any $u_1, u_2 \in U$ and for any integer n satisfying $0 \leq n < k$ it holds $u_1 \leq^n_{\mathbf{U}} u_2$ iff $f(u_1) \leq^n_{\mathbf{V}} f(u_2)$.

A *k-completion of a poset with tolerance (resp. equivalence)* $\mathbf{U}$ is an k-embedding $f : \mathbf{U} \to \mathbf{V}$ where $\mathbf{V}$ is a k-complete poset with tolerance (resp. equivalence). The k-completion $f : \mathbf{U} \to \mathbf{V}$ is called *minimal*, if for any other k-completion $f' : \mathbf{U} \to \mathbf{V}'$ there exists an k-embedding $h : \mathbf{V} \to \mathbf{V}'$ such that $f' = h \circ f$. For $k = \infty$, the k-completion of $\mathbf{U}$ is called simply *completion*.

The notion of a completion of a poset with tolerance seems to be new. Up to now, only various types of completions of ordered algebraic structures with operations have been studied [7].

6 Completions of Posets with Tolerance

Our goal in this section is to construct to a given poset with tolerance its minimal k-completion. We achieve this by identifying posets with tolerance with some fuzzy ordered sets and then performing the fuzzy version of the classical Dedekind-MacNeille completion of ordered sets.

We use basic notions from the theory of fuzzy ordered sets, as introduced in Sec. 3. However, we restrict ourselves only to three types of residuated lattices, namely the finite Łukasiewicz chain, three-element Gödel chain and countable Goguen chain. In this and the subsequent sections, by residuated lattice $\mathbf{L}$ we always mean a residuated lattice of one of these three types.

For a residuated lattice $\mathbf{L} = \langle L, \wedge, \vee, \otimes, \to, 0, 1 \rangle$ we say that $\mathbf{L}$ is a *finite Łukasiewicz chain*, if $L = \{a_0, a_1, \ldots, a_n\}$, $n > 1$, numbers $0 = a_0 < \cdots < a_n = 1$ are equidistant and $\otimes$ and $\to$ are given by (4); $\mathbf{L}$ is the *three-element Gödel chain*, if $L = \{0, 0.5, 1\}$ and $\otimes$ and $\to$ are given by (5); and $\mathbf{L}$ is the *countable Goguen chain*, if $L = \{2^{-n} \mid n = 0, 1, 2, \ldots\} \cup \{0\}$ and $\otimes$ and $\to$ are given by (6).

For each of these structures there is an element $g < 1$ such that for any $a < 1$ it holds $a \leq g$. For the finite Łukasiewicz chain we have $g = a_{n-1}$ and for the three-element Gödel chain as well as the countable Goguen chain we have $g = 0.5$. It holds $g^2 < g$ for the finite Łukasiewicz chain and the countable Goguen chain, while for the three-element Gödel chain we have $g^2 = g$. For the finite Łukasiewicz chain it holds $g^n = 0$ while for the other two structures $g^n > 0$ for any integer $n \geq 0$.

Let $\mathbf{U} = \langle U, \leq, \sim \rangle$ be a poset with tolerance. An $\mathbf{L}$-ordered set $\langle \langle U, \approx \rangle, \preceq \rangle$ is called an $\mathbf{L}$-*extension of* $\mathbf{U}$, if the following two conditions are satisfied:

1. ${}^1\preceq = \leq$,
2. ${}^g\approx = \sim$.

$\langle \langle U, \approx \rangle, \preceq \rangle$ is called *the minimal* $\mathbf{L}$-*extension of* $\mathbf{U}$, if for any other $\mathbf{L}$-extension $\langle \langle U, \approx' \rangle, \preceq' \rangle$ of $\mathbf{U}$ it holds $\preceq \subseteq \preceq'$. The minimal $\mathbf{L}$-extension of $\mathbf{U}$ is denoted $\mathbf{U_L}$.

Note that using the above conditions 1 and 2, $\mathbf{U}$ can be easily reconstructed from its $\mathbf{L}$-extension: $\mathbf{U} = \langle U, {}^1\preceq, {}^g\approx \rangle$. Thus, the minimal $\mathbf{L}$-extension $\mathbf{U_L}$ retains all the information about $\mathbf{U}$. For an $\mathbf{L}$-ordered set $\bar{\mathbf{U}} = \langle \langle U, \approx \rangle, \preceq \rangle$, the poset with tolerance $\langle U, {}^1\preceq, {}^g\approx \rangle$ is called *the g-reduction of* $\bar{\mathbf{U}}$.

Lemma 1. *The minimal* $\mathbf{L}$*-extension* $\mathbf{U_L} = \langle \langle U, \approx \rangle, \preceq \rangle$ *of* $\mathbf{U}$ *exists iff there is at least one* $\mathbf{L}$*-extension of* $\mathbf{U}$*. In such a case, if* $\{\langle \langle U, \approx_j \rangle, \preceq_j \rangle \mid j \in J\}$ *is the (non-empty) set of all* $\mathbf{L}$*-extensions of* $\mathbf{U}$ *then*

$$\preceq = \bigcap_{j \in J} \preceq_j .$$

Lemma 2. *If* $g^2 < g$*, then* $\mathbf{U_L}$ *exists iff* $\mathbf{U}$ *satisfies the diagonal property. If* $g^2 = g$*, then* $\mathbf{U_L}$ *exists iff* $\mathbf{U}$ *satisfies the strong diagonal property. In both cases, for any* $u_1, u_2 \in U$ *it holds* $u_1 \preceq u_2 = g^n$ *where* n *is the least element of* $\{0, 1, 2, \ldots\} \cup \{\infty\}$ *such that* $u_1 \leq^n_{\mathbf{U}} u_2$.

Proof. Suppose $\mathbf{U_L}$ exists and take $u_1, u_2 \in U$ such that $u_1 \leq^1_{\mathbf{U}} u_2$ and $u_2 \leq^1_{\mathbf{U}} u_1$. We have by definition of $\leq^1_{\mathbf{U}}$, (11), and transitivity of $\preceq$, that $u_1 \preceq u_2 \geq g$ and $u_2 \preceq u_1 \geq g$, and by antisymmetry of $\preceq$, $u_1 \approx u_2 \geq g$. Thus, $u_1 \sim u_2$. This shows that $\mathbf{U}$ satisfies the diagonal property.

Now suppose, in addition, that $g^2 = g$. Then for any $u_1, u_2 \in U$ such that $u_1 \leq^k_{\mathbf{U}} u_2$ and $u_2 \leq^k_{\mathbf{U}} u_1$ we have by definition of $\leq^1_{\mathbf{U}}$, (11), and transitivity of $\preceq$, $u_1 \preceq u_2 \geq g^k = g$ and $u_2 \preceq u_1 \geq g^k = g$, concluding again that $u_1 \sim u_2$, which proves that $\mathbf{U}$ satisfies the strong diagonal property.

To prove the opposite implication, we first set for any $u_1, u_2 \in U$, $u_1 \preceq u_2 = g^n$ where n is the least element of $\{0,1,2,\ldots\} \cup \{\infty\}$ such that $u_1 \leq^n_{\mathbf{U}} u_2$. We obtain a binary $\mathbf{L}$-relation $\preceq$ on U, which is reflexive, crisply antisymmetric and transitive. Indeed, reflexivity and crisp antisymmetry follow directly from the fact that $u_1 \preceq u_2 = 1$ iff $u_1 \leq u_2$. To prove transitivity, let $u_1, u_2, u_3 \in U$ and n_1 (resp. n_2) be the least element of $\{0,1,2,\ldots\} \cup \{\infty\}$ such that $u_1 \leq^{n_1}_{\mathbf{U}} u_2$ (resp. $u_2 \leq^{n_2}_{\mathbf{U}} u_3$). We have $(u_1 \preceq u_2) \otimes (u_2 \preceq u_3) = g^{n_1} \otimes g^{n_2} = g^{n_1+n_2}$. By definition, $u_1 \leq^{n_1+n_2}_{\mathbf{U}} u_3$, which proves transitivity.

Let $\approx$ be an $\mathbf{L}$-relation on U defined by (11). $\langle \langle U, \approx \rangle, \preceq \rangle$ is an $\mathbf{L}$-ordered set. We shall show that ${}^g{\approx} = {\sim}$. Let $u_1 \approx u_2 \geq g$. Then there exists an integer $k \geq 1$ such that $g^k = g$, $u_1 \leq^k_{\mathbf{U}} u_2$ and $u_2 \leq^k_{\mathbf{U}} u_1$. If $g^2 < g$, then we can set $k = 1$ and by the diagonal property, $u_1 \sim u_2$. If $g^2 = g$, then we use the strong diagonal property and obtain the same result. This shows ${}^g{\approx} \subseteq {\sim}$. Conversely, if $u_1 \sim u_2$, then $u_1 \leq^1_{\mathbf{U}} u_2$ and $u_2 \leq^1_{\mathbf{U}} u_1$, which, by definition of $\preceq$ and $\approx$, means $u_1 \approx u_2 \geq g$.

Theorem 3. *Let* $\mathbf{U}$, $\mathbf{V}$ *be two posets with tolerance,* $\mathbf{U_L}$, $\mathbf{V_L}$ *their* $\mathbf{L}$*-extensions, respectively,* $f : U \to V$ *a mapping. Further let* k *be the least element of* $\{1,2,\ldots\} \cup \{\infty\}$ *such that* $g^k = 0$*. Then* f *is a* k*-embedding of* $\mathbf{U}$ *into* $\mathbf{V}$*, if and only if it is an embedding of* $\mathbf{U_L}$ *into* $\mathbf{V_L}$*.*

Proof. Suppose f is a k-embedding of $\mathbf{U}$ into $\mathbf{V}$. By Lemma 2, for $u_1, u_2 \in U$ it holds $u_1 \preceq_{\mathbf{U}} u_2 = g^m$ where m is the least element of $\{0,1,2,\ldots\} \cup \{\infty\}$ such that $u_1 \leq^m_{\mathbf{U}} u_2$. Since f is a k-embedding then for any $n < k$ we have $f(u_1) \leq^n_{\mathbf{V}} f(u_2)$ iff $n \geq m$. Thus, if $m < k$, then $f(u_1) \preceq_{\mathbf{V}} f(u_2) = g^m$. If $m \geq k$, then $u_1 \preceq_{\mathbf{U}} u_2 = 0$ and there is no $n < k$ such that $f(u_1) \leq^n_{\mathbf{V}} f(u_2)$. From definition of $\preceq_{\mathbf{V}}$ we obtain $f(u_1) \preceq_{\mathbf{V}} f(u_2) \leq g^k = 0$.

To prove the opposite, we suppose that f is an embedding of $\mathbf{U_L}$ into $\mathbf{V_L}$. We need to show that for any integer n satisfying $0 \leq n < k$ it holds $u_1 \leq^n_{\mathbf{U}} u_2$, iff $f(u_1) \leq^n_{\mathbf{V}} f(u_2)$. Since $g^n > 0$ then $u_1 \leq^n_{\mathbf{U}} u_2$ (resp. $f(u_1) \leq^n_{\mathbf{V}} f(u_2)$) is equivalent to $u_1 \preceq_{\mathbf{U}} u_2 \geq g^n$ (resp. $f(u_1) \preceq_{\mathbf{V}} f(u_2) \geq g^n$). Thus, $(u_1 \preceq_{\mathbf{U}} u_2) = (f(u_1) \preceq_{\mathbf{V}} f(u_2))$.

Theorem 4. *If* k *is the least element of* $\{1,2,\ldots\} \cup \{\infty\}$ *such that* $g^k = 0$*, then any poset with tolerance* $\mathbf{U}$ *is* k*-complete, if and only if* $\mathbf{U_L}$ *exists and is a completely lattice* $\mathbf{L}$*-ordered set.*

Proof. Let $\langle U, \leq \rangle$ be a complete lattice and $\sim$ a complete tolerance. First we shall show that $\mathbf{U}$ satisfies the diagonal property. Let $u_1, u_2, u_3, u_4, v_1, v_2 \in U$ satisfy the assumption of the diagonal property. Since $v_1 = u_1 \wedge v_1$ and $v_2 = u_2 \wedge v_2$ then for $v_3 = u_3 \wedge v_2$ and $v_4 = u_4 \wedge v_1$ it holds $v_1 \sim v_4$ and $v_2 \sim v_3$. Now from $v_1 \vee v_3 = v_1$ and $v_2 \vee v_4 = v_2$ we obtain $v_1 \sim v_2$, proving the diagonal property.

Now let $g^2 < g$. According to Lemma 2, $\mathbf{U_L}$ exists. According to [8], to show that $\mathbf{U_L}$ is a completely lattice $\mathbf{L}$-ordered set it suffices to prove that for any $u \in U$ and $m \leq k$ there exists $v \in U$ such that for any $w \in U$ it holds

$$(w \preceq v) = g^m \to (w \preceq u). \tag{17}$$

The case $m = \infty$ is trivial as one can set $v = \bigvee U$ and obtain $(w \preceq v) = 1$, $g^m \to (w \preceq u) = 0 \to (w \preceq u) = 1$. In the rest of the proof we suppose $m < \infty$.

For any integer $n \geq 0$ denote by $u^{\sim^n}$ the greatest $u' \in U$ such that $u' \sim^n u$ (since $\mathbf{U}$ is k-complete then $u^{\sim^n}$ always exists). It can be easily proved by induction that $w \leq u^{\sim^m}$

but $w \not\leq u^{\sim^{m-1}}$. Set $v = u^{\sim^m}$ and denote $w \preceq u = g^n$ where $n < k$ or $n = k$, if $k < \infty$. Then n is the least integer such that $w \leq^n_{\mathbf{U}} u$.

If $n \leq m$, then $w \leq u^{\sim^n} \leq v$. Thus, $w \preceq v = 1$ and $g^m \to (w \preceq u) = g^m \to g^n = 1$, proving (17). If $m < n$, then $(w \preceq v) = g^{n-m} = g^m \to g^n$, proving (17) again.

The remaining case is $k = \infty$ and $w \preceq u = 0$. In this case, there is no integer n such that $w \leq^n_{\mathbf{U}} u$ which also means that there is no integer n' such that $w \leq^{n'}_{\mathbf{U}} v$. Thus, $(w \preceq v) = (w \preceq u) = 0$ and since for $k = \infty$ there are no zero divisors in $\mathbf{L}$, (17) is satisfied.

Now suppose $g^2 = g$. First, as it can be easily shown, since $\langle U, \leq \rangle$ is a complete lattice and $\sim$ a complete congruence then for any integer n from $u_1 \leq^n_{\mathbf{U}} u_2$ it follows $u_1 \leq^1_{\mathbf{U}} u_2$. Thus, from the fact that $\mathbf{U}$ satisfies the diagonal property (see the beginning of this proof) it follows that it satisfies the strong diagonal property and by Lemma 2, $\mathbf{U}_{\mathbf{L}}$ exists. The rest of the proof follows the same logic as the proof for $g^2 < g$.

Combining the results of Lemma 2 and Theorems 3, 4 we obtain the main result of this section.

Theorem 5. *1. Any poset with tolerance satisfying the diagonal property has the minimal k-completion for any $k \in \{1, 2, \ldots\} \cup \{\infty\}$.*

2. Any poset with equivalence satisfying the strong diagonal property has the minimal completion.

7 Tolerances Generated by Superrelations of Incidence Relation

The purpose of this section is to characterize all tolerance relations on concept lattices which can be generated by adding incidences to the incidence relation. We show that these tolerances are exactly those satisfying the diagonal property.

Let $\langle X, Y, I \rangle$ be a formal context, $\mathscr{B}(X, Y, I)$ its concept lattice. For a relation $J \supseteq I$ and $\langle A_1, B_1 \rangle, \langle A_2, B_2 \rangle \in \mathscr{B}(X, Y, I)$ we set $\langle A_1, B_1 \rangle \sim^J \langle A_2, B_2 \rangle$ iff $(A_1 \cup A_2) \times (B_1 \cup B_2) \subseteq J$. We obtain a tolerance $\sim^J$ on $\mathscr{B}(X, Y, I)$.

Conversely, for any tolerance relation $\sim$ on $\mathscr{B}(X, Y, I)$ we set

$$I^{\sim} = \bigcup_{\langle A_1, B_1 \rangle \sim \langle A_2, B_2 \rangle} (A_1 \cup A_2) \times (B_1 \cup B_2). \tag{18}$$

The definition of $I^{\sim}$ can be also reformulated as follows. It holds $\langle x, y \rangle \in I^{\sim}$ iff there exist $\langle A_1, B_1 \rangle, \langle A_2, B_2 \rangle \in \mathscr{B}(X, Y, I)$ such that $\langle \{x\}^{\uparrow\downarrow}, \{x\}^{\uparrow} \rangle \leq \langle A_1, B_1 \rangle, \langle A_1, B_1 \rangle \sim \langle A_2, B_2 \rangle$, and $\langle A_2, B_2 \rangle \leq \langle \{y\}^{\downarrow}, \{y\}^{\downarrow\uparrow} \rangle$.

A special type of superrelations of the incidence relation I is called a block relation [6]. A relation $J \subseteq X \times Y$, $J \supseteq I$ is a *block relation*, if for any object $x \in X$ the set $\{x\}^{\uparrow_J}$ is an intent of the formal context $\langle X, Y, I \rangle$ and for any attribute $y \in Y$, $\{y\}^{\downarrow_J}$ is an extent of $\langle X, Y, I \rangle$. Block relations are in one-to-one correspondence with complete tolerances on $\mathscr{B}(X, Y, I)$ [6]. This correspondence is given by the assignments $J \mapsto \sim^J$ and $\sim \mapsto I^{\sim}$ defined above.

Also, in the case that $\sim$ is a complete tolerance and J is a block relation, the factor lattice $\mathscr{B}(X, Y, I)/\sim^J$ is isomorphic to the concept lattice $\mathscr{B}(X, Y, J)$, which provides an efficient way of computing and representing the factor lattice. The isomorphism allows

us to use the concept lattice $\mathscr{B}(X,Y,J)$ instead of the (possibly much larger) concept lattice $\mathscr{B}(X,Y,I)$.

In the general case, when $\sim$ is not necessarily a complete tolerance and J is not a block relation, elements of $\mathscr{B}(X,Y,J)$ still represent some clusters of formal concepts from $\mathscr{B}(X,Y,I)$. More precisely, a formal concept $\langle A,B\rangle \in \mathscr{B}(X,Y,I)$ belongs to the maximal block represented by a formal concept $\langle A',B'\rangle \in \mathscr{B}(X,Y,J)$ iff $A \subseteq A'$ and $B \subseteq B'$. This shows that even in this general case it makes sense to use the concept lattice $\mathscr{B}(X,Y,J)$ as an approximate representation of $\mathscr{B}(X,Y,I)$.

The following theorem gives a general characterization of (not necessarily complete) tolerance relations on $\mathscr{B}(X,Y,I)$ generated by general superrelations $J \supseteq I$.

Theorem 6. *1. For any $J \supseteq I$, $\sim^J$ satisfies the diagonal property.*

2. The opposite always holds for the formal context $\langle U,U,\leq\rangle$: for any tolerance relation $\sim$ on a complete lattice **U** *satisfying the diagonal property there exist a superrelation $J \supseteq \leq$ such that $\sim^J = \sim$.*

Proof. 1. Suppose that the assumptions of the diagonal property are satisfied for $u_1 = \langle A_1,B_1\rangle$, $u_2 = \langle A_2,B_2\rangle$, $u_3 = \langle A_3,B_3\rangle$, $u_4 = \langle A_4,B_4\rangle$, $v_1 = \langle C_1,D_1\rangle$, $v_2 = \langle C_2,D_2\rangle$. We have $(C_1 \cup C_2) \times (D_1 \cup D_2) = (C_1 \times D_1) \cup (C_2 \times D_2) \cup (C_1 \times D_2) \cup (C_2 \times D_1)$, where, obviously, $C_1 \times D_1 \subseteq I \subseteq J$ and $C_2 \times D_2 \subseteq I \subseteq J$. Further, since $v_1 \leq u_1$, $v_2 \geq u_4$ and $u_1 \sim^J u_4$ then $C_1 \times D_2 \subseteq A_1 \times B_4 \subseteq (A_1 \cup A_4) \times (B_1 \cup B_4) \subseteq J$. Similarly, since $v_1 \geq u_3$, $v_2 \leq u_2$ and $u_3 \sim^J u_2$ then $C_2 \times D_1 \subseteq A_2 \times B_3 \subseteq (A_2 \cup A_3) \times (B_2 \cup B_3) \subseteq J$; hence $(C_1 \cup C_2) \times (D_1 \cup D_2) \subseteq J$.

2. Obviously, $\sim \subseteq \sim^J$. Suppose that $F(v_1) \sim^J F(v_2)$. Then $\langle v_1,v_2\rangle \in J$ and $\langle v_2,v_1\rangle \in J$. Since $J = I^{\sim}$ then from $\langle v_1,v_2\rangle \in J$ it follows that there are $u_1,u_4 \in U$ such that $u_1 \sim u_4$, $v_1 \leq u_1$, $v_2 \geq u_4$. Similarly, from $\langle v_2,v_1\rangle \in J$ it follows that there are $u_2,u_3 \in U$ such that $u_2 \sim u_3$, $v_2 \leq u_2$, $v_1 \geq u_3$. Now, from the diagonal property, $v_1 \sim v_2$.

8 Factorization of Concept Lattices

In this section, we use results of the previous sections to derive the main results of this paper. Basically, we show that factorization of a complete lattice by a (incompatible) tolerance consists of two steps: 1. constructing a k-completion of this lattice and, 2. factorizing the resulting complete lattice by a (now compatible) tolerance. We also show the correspondence of this general problem to the problem of a factorization of concept lattice by a superrelation of the incidence relation.

Let $\langle X,Y,I\rangle$ be a formal context, $J \supseteq I$ a superrelation of the incidence relation I, $\sim$ the tolerance relation on the concept lattice $\mathscr{B}(X,Y,I)$, induced by J (i.e., $\sim = \sim^J$). We also suppose that J is the minimal superrelation, given by $\sim$ (i.e., $J = I^{\sim}$). $\mathscr{B}(X,Y,I)$ together with $\sim$ forms a poset with tolerance, which we denote by $\langle \mathscr{B}(X,Y,I),\sim\rangle$.

By Theorem 6, $\sim$ satisfies the diagonal property. Thus, by Theorem 5, Part 1, there always exists a minimal k-completion of $\langle \mathscr{B}(X,Y,I),\sim\rangle$ for any $k \in \{1,2,\ldots\} \cup \{\infty\}$. If, in addition, $\sim$ is an equivalence and satisfies the strong diagonal property, then (by the same theorem, Part 2) there exists a minimal completion of $\langle \mathscr{B}(X,Y,I),\sim\rangle$ as a poset with equivalence.

In the following lemma, we construct a formal $\mathbf{L}$-context from $\langle X,Y,I\rangle$ in such a way that the resulting concept lattice is isomorphic to the minimal k-completion of the original concept lattice with tolerance $\sim$, $\langle\mathscr{B}(X,Y,I),\sim\rangle$. We achieve this by adding to the incidence relation I some values from $\mathbf{L}$ less than 1.

Let $I^J_{\mathbf{L}}: X\times Y\to L$ be an $\mathbf{L}$-relation between X and Y, defined by

$$I^J_{\mathbf{L}}(x,y) = g^n,$$

where $n\in\{0,1,2,\ldots\}\cup\{\infty\}$ is the least element such that

$$\langle\{x\}^{\uparrow_I\downarrow_I},\{x\}^{\uparrow_I}\rangle \leq^n_{\sim} \langle\{y\}^{\downarrow_I},\{y\}^{\downarrow_I\uparrow_I}\rangle.$$

Since the relation $\leq^1_{\sim}$ on $\mathscr{B}(X,Y,I)$ is equal to $\leq$ then for the 1-cut of the new incidence relation $I^J_{\mathbf{L}}$ it holds ${}^1I^J_{\mathbf{L}} = I$.

Lemma 3. *Let $f:\langle\mathscr{B}(X,Y,I),\sim\rangle\to\mathbf{V}$ be a minimal k-completion of $\langle\mathscr{B}(X,Y,I),\sim\rangle$ for some $k\in\{1,2,\ldots\}\cup\{\infty\}$. If $g^2<g$ holds in $\mathbf{L}$ and k is the least element satisfying $g^k=0$, then $\mathbf{V}_{\mathbf{L}}$ is isomorphic to $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$. If $\sim$ is an equivalence satisfying the strong diagonal property and f a minimal completion of $\langle\mathscr{B}(X,Y,I),\sim\rangle$ as a poset with equivalence, then the same also holds for $g^2=g$.*

Consequently, $\mathbf{V}$ is in both cases isomorphic to the g-reduction of $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$ and, using this isomorphism, the complete tolerance $\sim_{\mathbf{V}}$ corresponds to the g-cut ${}^g\approx$ of the $\mathbf{L}$-equality $\approx$ on $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$.

Proof. By definition of $I^J_{\mathbf{L}}$ and Lemma 2, we have for any $x\in X$ and $y\in Y$,

$$I^J_{\mathbf{L}}(x,y) = \langle\{x\}^{\uparrow_I\downarrow_I},\{x\}^{\uparrow_I}\rangle \preceq_{\mathbf{V}_{\mathbf{L}}} \langle\{y\}^{\downarrow_I},\{y\}^{\downarrow_I\uparrow_I}\rangle.$$

Thus, we can set

$$\gamma(x) = f(\langle\{x\}^{\uparrow_I\downarrow_I},\{x\}^{\uparrow_I}\rangle),\quad \mu(y) = f(\langle\{y\}^{\downarrow_I},\{y\}^{\downarrow_I\uparrow_I}\rangle)$$

and use Theorem 1.

In the case $g^2=0$ (i.e., $\mathbf{L}$ is the three-element Łukasiewicz chain), the $\mathbf{L}$-relation $I^J_{\mathbf{L}}$ can be easily constructed. Namely, it holds

$$I^J_{\mathbf{L}}(x,y) = \begin{cases} 1 & \text{if } \langle x,y\rangle\in I,\\ 0.5 & \text{if } \langle x,y\rangle\in J\setminus I,\\ 0 & \text{otherwise.}\end{cases} \tag{19}$$

In this case, we have the following result:

Theorem 7. *Let $\mathbf{V}$ be a 2-completion of $\langle\mathscr{B}(X,Y,I),\sim\rangle$. Then $\mathscr{B}(X,Y,J)$ is isomorphic to $\mathbf{V}/{\sim_{\mathbf{V}}}$.*

Proof. Let $\mathbf{L}$ be the three-element Łukasiewicz chain. According to Lemma 3, $\mathbf{V}_{\mathbf{L}}$ is isomorphic to the $\mathbf{L}$-concept lattice $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$ where $I^J_{\mathbf{L}}$ is given by (19). For $e=0.5$, the factor residuated lattice $\mathbf{L}/e$ is isomorphic to the 2-element Boolean algebra $\mathbf{2}$, where $[0]^{0.5}=0\in L/0.5$ and $[0.5]^{0.5}=[1]^{0.5}=1\in L/0.5$. Thus, $[I^J_{\mathbf{L}}]^{0.5}(x,y)=1$ iff $\langle x,y\rangle\in J$, and the $\mathbf{L}/e$-concept lattice $\mathscr{B}(X,Y,[I^J_{\mathbf{L}}]^{0.5})$ is isomorphic as an ordered set with the concept lattice $\mathscr{B}(X,Y,J)$. Now we can use Theorem 2.

Recall that if J is a block relation (and, consequently, $\sim$ is a complete tolerance), the factor lattice $\mathscr{B}(X,Y,I)/{\sim^J}$ is isomorphic to the concept lattice $\mathscr{B}(X,Y,J)$. The above theorem clarifies the correspondence between these two sets in the general case. The factorization is achieved in two steps. First, embedding the concept lattice $\mathscr{B}(X,Y,I)$ into its 2-completion $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$ in such a way that the tolerance $\sim$ is a restriction of the 0.5-cut ${}^{0.5}{\approx}$ of the $\mathbf{L}$-equality $\approx$ on $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$. Second, factorizing the lattice $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$ by the complete tolerance ${}^{0.5}{\approx}$.

Since the concept lattice $\mathscr{B}(X,Y,I)$ is embedded into its 2-completion $\mathscr{B}(X,Y,I^J_{\mathbf{L}})$ before factorization, it is possible that there are formal concepts in $\mathscr{B}(X,Y,J)$ which do not correspond to any maximal block of the tolerance $\sim$ and, as such, do not belong to the factor set $\mathscr{B}(X,Y,I)/{\sim^J}$. In the next theorem we show that this is not the case when $\sim$ is an equivalence satisfying the strong diagonal property.

Theorem 8. *If $\sim$ is an equivalence satisfying the strong diagonal property, then for any formal concept $\langle A,B\rangle \in \mathscr{B}(X,Y,I)$ there exists exactly one formal concept $\langle A',B'\rangle \in \mathscr{B}(X,Y,J)$ such that $A \subseteq A'$ and $B \subseteq B'$. The induced mapping $\mathscr{B}(X,Y,I) \to \mathscr{B}(X,Y,J)$ is surjective.*

Proof. This follows from the following observation. If $\mathbf{U}$ is an $\mathbf{L}$-ordered set such that $\langle U, {}^1{\preceq_{\mathbf{U}}}\rangle$ is a complete lattice (${}^1{\preceq_{\mathbf{U}}}$ is the 1-cut of $\preceq_{\mathbf{U}}$), then the Dedekind-MacNeille completion $f\colon \mathbf{U} \to \mathbf{V}$ adds to $\mathbf{U}$ only elements similar to existing elements to the degree g. More precisely, for any $v \in V$ there exists $u \in U$ such that $(v \approx_{\mathbf{V}} f(u)) \geq g$.

This result is a little surprising. It says that if an equivalence on a complete lattice satisfies the strong diagonal property, then the factor set is a complete lattice even though the equivalence is not a congruence.

9 Conclusion

This paper provides a theoretical background for the problem of factorization of concept lattices by incompatible tolerances. This problem arises naturally in situations where we are trying to reduce the size of a concept lattice based on a similarity of concepts, which is not given by a block relation and, as such, is not compatible with the complete lattice structure of the concept lattice.

We also introduced and solved a new problem of finding a minimal completion of a poset with tolerance and poset with equivalence.

We used the theory of fuzzy ordered sets and fuzzy concept lattices as the main tool in the paper. Our results, however, belong to the realm of classical (i.e., not fuzzy) ordered sets and concept lattices. As it turns out, various choices of the structure of truth degrees allow us to prove various interesting results. More specifically, the countable Goguen chain has been used to find the minimal completion of a poset with tolerance, the three-element Gödel chain allowed similar results for posets with equivalence. We used the three-element Łukasiewicz chain and the three-element Gödel chain to prove the main results of the paper (Theorem 7, 8).

The theoretical results, presented in this paper, could be used in situations where we deal with tolerances on concept lattices which are not complete. We mention two examples.

In [4], the use of concept lattices for visualizing results of some information retrieval tasks (namely, term-document relationships) is proposed. The authors attempt to reduce the size of concept lattices (to make them more comprehensible for the user) by using Singular Value Decomposition to introduce an equivalence $\sim_Y$ on the set Y of attributes and then computing the factor set $\mathscr{B}(X,Y,I)/\sim$, where $\sim$ is an equivalence, induced on $\mathscr{B}(X,Y,I)$ by $\sim_Y$. The authors claim that $\sim$ is a congruence, which, in fact, is not true. However, it is possible to use the results of this paper to partially overcome this obstacle: It can be easily shown that $\sim$ can be defined by means of some superrelation $J \supseteq I$ and satisfies the strong diagonal property. Thus, according to Theorem 8, there is a (surjective) factor projection $\mathscr{B}(X,Y,I) \to \mathscr{B}(X,Y,J)$ (which is not, however, a homomorphism of lattices) and a bijection between $\mathscr{B}(X,Y,J)$ and $\mathscr{B}(X,Y,I)/\sim$.

Paper [10] is an overview of several applications of concept lattices in software analysis. In Conclusion, the author admits that "... all these applications stick to the basic theory of concept lattices and their corresponding implication base, but do not apply more advanced results ... The reason is that realistic lattices do not have the properties required for the advanced techniques. For examples, typical lattices in software technology have neither congruences nor block relations ..." The results of the present paper could be used in these situations.

In addition to possible applications, there are several new theoretical problems which could be addressed:

- the general meaning of the choice of different structures of truth degrees,
- generalizing results of this paper to fuzzy FCA,
- the problem of superfluous concepts in $\mathscr{B}(X,Y,J)$ when $\sim$ is not an equivalence.

Some of these problems will be addressed in a forthcoming paper.

References

1. Belohlavek, R.: Concept lattices and order in fuzzy logic. Ann. Pure Appl. Log. 128(1-3), 277–298 (2004)
2. Belohlavek, R.: Similarity relations in concept lattices. J. Log. Comput. 10(6), 823–845 (2000)
3. Belohlavek, R.: Fuzzy Relational Systems: Foundations and Principles. Kluwer Academic Publishers, Norwell (2002)
4. Cheung, K., Vogel, D.: Complexity reduction in lattice-based information retrieval. Inf. Retr. 8(2), 285–299 (2005)
5. Czédli, G.: Factor lattices by tolerances. Acta Sci. Math. 44, 35–42 (1982)
6. Ganter, B., Wille, R.: Formal Concept Analysis – Mathematical Foundations. Springer, Heidelberg (1999)
7. Harding, J.: Completions of ordered algebraic structures: A survey. In: Huynh, V.N., et al. (eds.) Interval / Probabilistic Uncertainty and Non-Classical Logics. AISC, vol. 46, pp. 231–244. Springer, Heidelberg (2008)

8. Krupka, M.: An alternative version of the main theorem of fuzzy concept lattices. In: Trappl, R. (ed.) Cybernetics and Systems 2010, pp. 9–14. Austrian Society for Cybernetic Studies, Vienna (2010) (extended version submitted)
9. Krupka, M.: Factorization of residuated lattices. Log. J. IGPL 17(2), 205–223 (2009)
10. Snelting, G.: Concept lattices in software analysis. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 272–287. Springer, Heidelberg (2005)
11. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered Sets, Boston, pp. 445–470 (1982); seminal publication on Formal Concept Analysis
12. Wille, R.: Complete tolerance relations of concept lattices. In: Eigenthaler, G., et al. (eds.) Contributions to General Algebra, vol. 3, pp. 397–415. Hölder-Pichler-Tempsky, Wien (1985)

Merging Ordered Sets

Bernhard Ganter[1], Christian Meschke[1,⋆], and Henri Mühle[2,⋆⋆]

[1] Institut für Algebra, TU Dresden, Germany
{bernhard.ganter,christian.meschke}@tu-dresden.de
[2] Fakultät für Mathematik, Universität Wien, Austria
henri.muehle@univie.ac.at

Abstract. While extending partial orders towards linear orders is a very well-researched topic, the combination of two ordered sets has not yet attracted too much attention. In the underlying article, however, we describe the possibilities to merge two given quasiordered sets in the sense that the restriction of the combined order towards the given ordered sets returns the initial orders again. It turns out that these *mergings* form a complete lattice. We elaborate these lattices of mergings and present its contextual representation. While the motivating example was discovered in role-oriented software modeling, we give a further possible application in the field of scheduling.

1 Introduction

In order theory, a well-studied problem is the question of finding linear extensions of a given partial order. In this paper we will investigate the somehow related problem of *merging* two given orders $(P, \leq_P)$ and $(Q, \leq_Q)$. Thereby, we understand a merging as an order on $P \cup Q$, such that the restrictions onto P and Q return the initial posets again.

Such a construction can for example be observed, when considering the role-play relation in role-oriented software modeling. We refer to [Ste00] for a detailed introduction into this topic and will recall only the immediate notations. Roughly spoken, a role-oriented software model consists of two independent hierarchies $(P, \leq_P)$ and $(Q, \leq_Q)$ – called *base* resp. *role* types – and a partial order $R \subseteq P \times Q$, that determines which role type can be played by which base type. Since $(P, \leq_P)$ and $(Q, \leq_Q)$ can be regarded as a kind of generalisation/specialisation hierarchy the role-play property is inherited according to the following restriction [Ste00, p. 13]: Given two base types $p' \leq_P p$ from P and two role types $q \leq_Q q'$ from Q it has to follow that $pRq \Rightarrow p'Rq'$. Therefore, for fixed $p \in P, q \in Q$ with pRq, we obtain that the set

$$p^R := \{q' \in Q \mid pRq'\} \quad \text{resp.} \quad q^R := \{p' \in P \mid p'Rq\}$$

is an order filter in $(Q, \leq_Q)$ resp. an order ideal in $(P, \leq_P)$. We shall note that this already means that a proper role-play relation always forms a bond between

⋆ Supported by the DFG research grant no. GA 216/10-1.
⋆⋆ Supported by the FWF research grant no. Z130-N13.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 183–203, 2011.

$(P, P, \not\geq_P)$ and $(Q, Q, \not\geq_Q)$. One may use this property for developing a design advisor for creating role models, which, however, is beyond the scope of the present article. Furthermore, the relation $\leq_R := \leq_P \cup \leq_Q \cup R$ clearly is a partial order on $P \cup Q$ whose restrictions on P resp. Q agree with $\leq_P$ resp. $\leq_Q$. (This is clear, since there are no pairs of the form (q, p) with $q \in Q, p \in P$ and thus the transitivity of $\leq_R$ does not extend P resp. Q.) From this motivating example we started to investigate the mergings

$$\leq_{R,S} := \leq_P \cup \leq_Q \cup R \cup S$$

between two given ordered sets $(P, \leq_P)$ and $(Q, \leq_Q)$ where $R \subseteq P \times Q$ and $S \subseteq Q \times P$.

The rest of this paper is structured as follows: In Section 2 we prepare the basic definitions. In Section 3 we introduce our notion of a (proper) merging and discuss fundamental properties. Section 4 is dedicated to the contextual representation of the lattice of (proper) mergings for two given posets. Section 5 illustrates our construction and gives a possible application in the field of scheduling. Section 6 concludes this paper and gives an outlook towards future work.

2 Preliminaries

We suppose that the reader is familiar with the basic concepts of order theory. Nevertheless we shortly repeat some of the most fundamental notions in the following. For a detailed introduction to this topic we refer the reader to [DP02]. It is common usage to call a set P equipped with a reflexive and transitive relation $\leq$ on it a **quasiordered set**. We write $x < y$ for $x \leq y$ and $x \not\geq y$. For a given quasiordered set $\mathbb{P} = (P, \leq)$ one obtains an equivalence relation $\equiv$ on P by

$$x \equiv y \ :\Longleftrightarrow \ x \leq y \text{ and } y \leq x.$$

The equivalence $\equiv$ equals the identity relation on P if and only if the quasiorder relation $\leq$ is a **(partial) order** relation on P. This means that the relation $\leq$ is also antisymmetric. It is commonly known that the relation $\trianglelefteq$ defined by

$$[x]_\equiv \trianglelefteq [y]_\equiv \ :\Longleftrightarrow \ x \leq y,$$

is an order relation on the set of all $\equiv$-equivalence classes. The partially ordered set

$$\mathbb{P}^* := (P/\equiv, \trianglelefteq)$$

is then called the **factor poset** of the quasiordered set $\mathbb{P}$.

If there is no ambiguity about the quasiorder relation, we allow to omit it in our notation and to simply speak of the quasiordered set P. A subset X of the quasiordered set P is called an **order ideal** if for every $x \in X$ and every $p \in P$ with $p \leq x$ it follows that $p \in X$. The **dual** of $(P, \leq)$ is the quasiordered set $(P, \leq)^d := (P, \geq)$, where $x \geq y$ simply means that $y \leq x$. Furthermore, $X \subseteq P$ is called an **order filter** in P if it is an order ideal in its dual P^d.

There is a nice way to describe order ideals and order filters of an ordered set $\mathbb{P}$ by means of Formal Concept Analysis (FCA). The basic elements of FCA are **formal context**s, i. e. triplets (G, M, I) where G is a set of **objects**, M is a set of **attributes** and $I \subseteq G \times M$ is a relation describing whether an object **has** an attribute. From this context we can now derive **formal concepts** via the derivation operators

$$(\cdot)^I : \mathfrak{P}(G) \to \mathfrak{P}(M), A \mapsto \{m \in M \mid \forall\, g \in A : gIm\},$$
$$(\cdot)^I : \mathfrak{P}(M) \to \mathfrak{P}(G), B \mapsto \{g \in G \mid \forall\, m \in B : gIm\}.$$

A formal concept is a pair (A, B) with $A \subseteq G, B \subseteq M$ satisfying $A^I = B$ and $B^I = A$. A is called the **extent**, B is called the **intent** of this concept. There is a natural ordering on the set $\mathfrak{B}(G, M, I)$ of concepts given by

$$(A_1, B_1) \leq (A_2, B_2) :\Longleftrightarrow A_1 \subseteq A_2 \quad (\Leftrightarrow B_1 \supseteq B_2)$$

such that the ordered set $\underline{\mathfrak{B}}(G, M, I) := (\mathfrak{B}(G, M, I), \leq)$ is a complete lattice – the **concept lattice** of (G, M, I) [GW99, p. 20]. For $g \in G$ and $m \in M$ we put $g^I := \{g\}^I$ and $m^I := \{m\}^I$.

For a given quasiordered set $(P, \leq_P)$ there is a formal context whose extents correspond to the order ideals of P and whose intents correspond to the order filters of P – the **contraordinal scale** $(P, P, \not\geq_P)$ [GW99, p. 49]. The concepts of the contraordinal scale are precisely the pairs of the form $(A, P \setminus A)$ where A is an order ideal of P. One can furthermore show that the contraordinal scales precisely describe the doubly-founded, completely distributive lattices [GW99, p. 49].

Let (G, M, I) and (H, N, J) be formal contexts. One calls a relation $R \subseteq G \times N$ a **bond** from (G, M, I) to (H, N, J) if for every object $g \in G$ the row g^R is an intent of (H, N, J) and for every attribute $n \in N$ the column n^R is an extent of (G, M, I). A bond from (G, M, I) to (G, M, I) is called a **self-bond**.

3 Mergings and Proper Mergings

In the following $(P, \leq_P)$ and $(Q, \leq_Q)$ denote two quasiordered sets. We assume P and Q to be disjoint, which allows us to simply write $x \leq y$ instead of $x \leq_P y$ or $x \leq_Q y$, respectively. The **cardinal sum** of $(P, \leq_P)$ and $(Q, \leq_Q)$ is the quasiordered set

$$(P, \leq_P) + (Q, \leq_Q) := (P \cup Q, \leq_P \cup \leq_Q).$$

Hence, two elements are comparable in the cardinal sum if and only if they belong to the same component and are comparable in this component.

Definition 1. *A pair (R, S) where $R \subseteq P \times Q$ and $S \subseteq Q \times P$ is called a* **merging** *of the disjoint quasiordered sets $(P, \leq_P)$ and $(Q, \leq_Q)$ if the relation*

$$\leq_{R,S} := \bigcup\{\leq_P, \leq_Q, R, S\}$$

is a quasiorder on $P \cup Q$. A merging (R, S) is called **proper** *if $R \cap S^{-1}$ is empty. The set of all mergings from P to Q is denoted by $\mathfrak{M}_{P,Q}$, or simply by $\mathfrak{M}$. The set of all proper mergings is denoted by $\mathfrak{M}^{\bullet}_{P,Q}$, or simply by $\mathfrak{M}^{\bullet}$.*

Obviously, $p\, R\, q$ means that the element p from P is less or equal than q from Q with respect to the quasiorder $\leq_{R,S}$ on $P \cup Q$. Dually, for $p \in P$ and $q \in Q$ $q\, S\, p$ expresses that $q \leq_{R,S} p$ is satisfied. Hence, a merging is not proper if and only if there are elements $p \in P$ and $q \in Q$ with

$$p \leq_{R,S} q \quad \text{and} \quad q \leq_{R,S} p.$$

	P	Q
P	$\leq_P$	R
Q	S	$\leq_Q$

In other words the pairs (p, q) from $R \cap S^{-1}$ can be understood as a fusion of p and q. They belong to the same equivalence class in the factor poset of $(P \cup Q, \leq_{R,S})$.

Remark 1. The pair (R, S) is a (proper) merging of the quasiordered sets P and Q if and only if the pair $(\hat{R}, \hat{S})$ with

$$\hat{R} := \{([p]_{\equiv}, [q]_{\equiv}) \mid (p, q) \in R\} \quad \text{and} \quad \hat{S} := \{([q]_{\equiv}, [p]_{\equiv}) \mid (q, p) \in S\}$$

is a (proper) merging of the factor posets P^* and Q^*. Furthermore, every (proper) merging of P^* and Q^* is of this form. Hence, if one wants to describe (proper) mergings of quasiordered sets, this one-to-one correspondence allows to describe the (proper) mergings of the factor posets instead.

Proposition 1. *Let $(P, \leq_P)$ and $(Q, \leq_Q)$ be quasiordered sets and let $R \subseteq P \times Q$. Then the following three statements are equivalent:*

(a) For every $p \in P$ the row p^R is an order filter in Q, and for every $q \in Q$ the column q^R is an order ideal in P;

(b) R is an order ideal in the quasiordered set $P \times Q^d$;

(c) R is a bond from $(P, P, \not\geq_P)$ to $(Q, Q, \not\geq_Q)$.

Proof. We initiate our proof by defining for $(p_1, q_1), (p_2, q_2) \in P \times Q$

$$(p_1, q_1) \sqsubseteq (p_2, q_2) \;:\Longleftrightarrow\; p_1 \leq p_2 \text{ and } q_1 \geq q_2.$$

Hence, $\sqsubseteq$ is the quasiorder relation of $P \times Q^d$. Now suppose that (a) is valid. Let $(x, y) \in R$ and $(p, q) \in P \times Q$ with $(p, q) \sqsubseteq (x, y)$. This means that $p \leq x\, R\, y \leq q$. Since by (a) $x^R = \{\hat{q} \in Q \mid x\, R\, \hat{q}\}$ is an order filter in Q, we infer that $x\, R\, q$, and since by (a) q^R is an order ideal in P, we infer from $x \in q^R$ that $p\, R\, q$. Hence, (a) implies (b). Now suppose that (b) is valid and let $p \in P$, $q \in p^R$ and $y \in Q$ with $q \leq y$. This means that $(p, q) \in R$ and $(p, y) \sqsubseteq (p, q)$. From (b) we infer that $y \in p^R$. Hence, p^R is an order filter in Q. Dually, (b) implies that q^R is an order ideal in P for every $q \in Q$.

The equivalence of (a) and (c) easily follows from the fact that the intents of the contraordinal scale $(P, P, \not\geq_P)$ are precisely the order filters of P and that the extents of the contraordinal scale $(Q, Q, \not\geq_Q)$ are precisely the order ideals of Q. □

Proposition 2. *Let $(P, \leq_P)$ and $(Q, \leq_Q)$ be disjoint quasiordered sets, let $R \subseteq P \times Q$ and let $S \subseteq Q \times P$. Then the pair (R, S) is a merging if and only if all of the following four properties are satisfied:*

(1) R is an order ideal in $P \times Q^d$,

(2) S^{-1} is an order filter in[1] $P \times Q^d$,

(3) $R \circ S \subseteq \leq_P$,

(4) $S \circ R \subseteq \leq_Q$.

Furthermore, $\leq_{R,S}$ is antisymmetric iff both, $\leq_P$ and $\leq_Q$ are antisymmetric and the intersection $R \cap S^{-1}$ is empty.

Proof. We have to show that $\leq_{R,S}$ is a quasiorder on $P \cup Q$ iff all of the four properties (1) to (4) are satisfied. Let us assume that (1) to (4) are valid. Since $\leq_{R,S}$ is reflexive for trivial reasons, it is enough to show transitivity. Let $x, y, z \in P \cup Q$ with $x \leq_{R,S} y \leq_{R,S} z$. We have to show that this implies $x \leq_{R,S} z$, which is obviously true when all three elements belong to P or all three belong to Q. In the case that $x, y \in P$ and $z \in Q$ we have that $x \leq y \; R \; z$. By (1) and Proposition 1 we infer that $x \; R \; z$ and hence $x \leq_{R,S} z$. Analogously, the three cases $x \in Q$, $y, z \in P$, and $x, y \in Q$, $z \in P$, and $x \in P$, $y, z \in Q$ imply $x \leq_{R,S} z$. In the case $x, z \in P$ and $y \in Q$, we infer $x \; R \; y \; S \; z$ and hence $(x, z) \in R \circ S$. By (3) we infer that $x \leq z$ and hence $x \leq_{R,S} z$. Dually, in the case $x, z \in P$ and $y \in P$ property (4) yields $x \leq_{R,S} z$. The backward direction, i.e. that the transitivity of $\leq_{R,S}$ implies (1) to (4), is even easier and hence omitted.

That the antisymmetry of $\leq_{R,S}$ implies that of $\leq_P$ and of $\leq_Q$ is trivial. If a pair (p, q) belongs to $R \cap S^{-1}$, this just means that $p \leq_{R,S} q$ and $q \leq_{R,S} p$. In the case of antisymmetry of $\leq_{R,S}$ this yields $p = q$, which contradicts the disjointness of P and Q. That, otherwise, $R \cap S^{-1} = \emptyset$ and the antisymmetry of $\leq_P$ and $\leq_Q$ imply the antisymmetry of $\leq_{R,S}$ can be shown by an obvious case-by-case analysis. □

Corollary 1. *Let P and Q be disjoint partially ordered sets, let $R \subseteq P \times Q$ and let $S \subseteq Q \times P$. Then $(P \cup Q, \leq_{R,S})$ is a partially ordered set again if and only if (R, S) is a proper merging.*

Hence, if one just considers posets and wants to merge them to a new poset, the notion of a proper merging seems to be the right one. Because in this case one has to avoid to fuse elements from P with elements from Q. In the following we will see that both concepts, the mergings and the proper mergings, can be described in a similar fashion. We will thus often have propositions that are divided into two parts: one considering mergings, the other one considering proper mergings.

[1] In order to avoid the somehow annoying inversion of S, we will often use the equivalent condition that S is an order filter in $Q^d \times P$.

3.1 Lattices of Mergings

We will see that the (proper) mergings form a complete lattice in a canonical manner. In order to prepare this statement we need the following Lemma 1 saying that the relation product $\circ$ respects the union $\bigcup$ of relations.

Lemma 1. *Let A, B, C be sets, let $X \subseteq A \times B$ and let $Y_t \subseteq B \times C$ for $t \in T$. Then*

$$X \circ \bigcup_{t \in T} Y_t \;=\; \bigcup_{t \in T} (X \circ Y_t)\,.$$

Proof. We omit the proof since it is common knowledge. □

Theorem 1. *Let P and Q be disjoint quasiordered sets.*

(i) Then the set $\mathfrak{M}$ of all mergings of P and Q forms a complete lattice if one orders it by

$$(R_1, S_1) \leq (R_2, S_2) \;:\Longleftrightarrow\; R_1 \subseteq R_2 \text{ and } S_1 \supseteq S_2.$$

The indicated expressions for infimum and supremum are given by:

$$\bigwedge_{t \in T} (R_t, S_t) \;=\; \Big(\bigcap_{t \in T} R_t, \bigcup_{t \in T} S_t\Big),$$

$$\bigvee_{t \in T} (R_t, S_t) \;=\; \Big(\bigcup_{t \in T} R_t, \bigcap_{t \in T} S_t\Big).$$

(ii) The set $\mathfrak{M}^\bullet$ of all proper mergings forms a complete sublattice of $\mathfrak{M}$.

(iii) The least (proper) merging is $(\emptyset, Q \times P)$, whereas the greatest one is $(P \times Q, \emptyset)$.

Proof. In order to prove (i) it suffices to show that the indicated infimum and supremum are well-defined, i.e., that both constructions yield to a merging again. Afterwards it is obvious that this indeed is the infimum and supremum in $(\mathfrak{M}, \leq)$. Let for every $t \in T$ the pair (R_t, S_t) be a merging. Since order ideals and order filters are closed under both, set union and set intersection, one easily infers from Proposition 2 that $R := \bigcap_{t \in T} R_t$ is an order ideal in $P \times Q^d$, and that $S := \bigcup_{t \in T} S_t$ is an order filter in $Q^d \times P$ again. Furthermore, it follows that

$$R \circ S \;=\; \Big(\bigcap_{s \in T} R_s\Big) \circ \Big(\bigcup_{t \in T} S_t\Big) \;=\; \bigcup_{t \in T} \Big(\Big(\bigcap_{s \in T} R_s\Big) \circ S_t\Big) \;\subseteq\; \bigcup_{t \in T} R_t \circ S_t \;\subseteq\; \leq_P .$$

Thereby the second equality follows from the aforementioned Lemma 1 and the first inclusion follows from the monotonicity of the relation product $\circ$. By Proposition 2 we have shown that (R, S) is indeed a merging. Dually one can show that the given supremum is a merging again.

We now show that the proper mergings form a complete sublattice. Let therefore the pair (R_t, S_t) be a proper merging for every $t \in T$. This implies that

$$\Big(\bigcap_{s\in T} R_s\Big) \cap \Big(\bigcup_{t\in T} S_t\Big)^{-1} = \Big(\bigcap_{s\in T} R_s\Big) \cap \Big(\bigcup_{t\in T} S_t^{-1}\Big) = \bigcup_{t\in T}\Big(S_t^{-1} \cap \bigcap_{s\in T} R_s\Big) = \bigcup_{t\in T} \emptyset = \emptyset.$$

Dually one can show that $\mathfrak{M}^\bullet$ is closed under suprema in $\mathfrak{M}$. Statement (iii) is an easy consequence of (i) and (ii). □

As a consequence of the previous theorem we obtain the following. Let R be an order ideal in $P \times Q^d$. By Proposition 2, R is a possible first component of a merging. The question if there really is an S such that (R, S) is a merging is easy to answer, because $(R, \emptyset)$ obviously is a merging. Furthermore, $(R, \emptyset)$ is the largest merging having R as its first component. And since R is the first component of a merging, Theorem 1 also implies that

$$\bigwedge\{(R, \tilde{S}) \mid \tilde{S} \subseteq Q \times P \text{ such that } (R, \tilde{S}) \text{ merging}\}$$

is the smallest merging with R as its first component. Hence, if one fixes the first component one obtains an interval in $\mathfrak{M}$. A similar consequence is the following

Corollary 2. *Let P and Q be disjoint quasiordered sets, let $X \subseteq P \times Q$ and let $Y \subseteq Q \times P$. Then the set of all (proper) mergings (R, S) with $X \subseteq R$ and $Y \subseteq S$ is empty or forms an interval in $\mathfrak{M}$ (in $\mathfrak{M}^\bullet$).*

For two mergings (R_1, S_1) and (R_2, S_2) we define

$$(R_1, S_1) \subseteq^2 (R_2, S_2) \;:\Longleftrightarrow\; R_1 \subseteq R_2 \text{ and } S_1 \subseteq S_2.$$

Obviously, $\subseteq^2$ defines an order relation on $\mathfrak{M}$. In case of $(R_1, S_1) \subseteq^2 (R_2, S_2)$ we say that (R_2, S_2) is an **extension** of (R_1, S_1). Hence, (R_2, S_2) is an extension of (R_1, S_1) if and only if $\leq_{R_2,S_2}$ is an order extension of $\leq_{R_1,S_1}$, i.e., if $\leq_{R_1,S_1} \subseteq \leq_{R_2,S_2}$. Obviously, the merging $(\emptyset, \emptyset)$ which corresponds to the cardinal sum of P and Q is the least element of the ordered set $(\mathfrak{M}, \subseteq^2)$, as well as of $(\mathfrak{M}^\bullet, \subseteq^2)$. The maximal elements of $(\mathfrak{M}, \subseteq^2)$ are called **maximal** mergings and the maximal elements of $(\mathfrak{M}^\bullet, \subseteq^2)$ are called **maximal proper** mergings. Please note, that the latter notion is capable of being misunderstood. A merging that is proper and maximal is of course maximal proper, but not every maximal proper merging needs to be maximal in $(\mathfrak{M}, \subseteq^2)$.

3.2 An Example

Let us illustrate the previous constructions by means of the two two-element chains $(P, \leq_P)$ and $(Q, \leq_Q)$ as shown in Fig. 1. Let us for this setting consider some example mergings. In Fig. 2(a) we clearly have $R_1 = \emptyset, S_1 = \emptyset$.

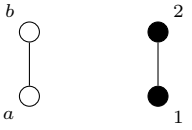
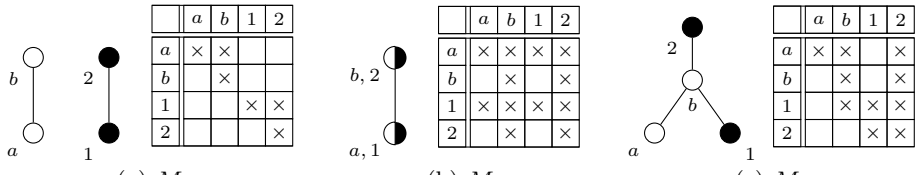


Fig. 1. Two chains $(P, \leq_P)$, $(Q, \leq_Q)$

This merging is proper and corresponds to the cardinal sum of $(P, \leq_P)$ and $(Q, \leq_Q)$. It introduces no additional order relationships on $P \cup Q$. Fig. 2(b), however, shows no proper merging. We can read

$$R_2 = \{(a,1),(a,2),(b,2)\},$$
$$S_2 = \{(1,a),(1,b),(2,b)\}.$$

Thus, it is $R \cap S^{-1} = \{(a,1),(b,2)\}$. A nontrivial proper merging is e. g. given in Fig. 2(c). We have $R_3 = \{(a,2),(b,2)\}, S_3 = \{(1,b)\}$ and thus $R \cap S^{-1} = \emptyset$ which makes this merging proper. Comparing the proper mergings M_1 and M_3, we see that $R_1 \subsetneq R_3$ and $S_1 \subsetneq S_3$ which makes them incomparable w.r.t. the lattice order $\leq$ on $\mathfrak{M}^\bullet$. But clearly, M_2 and M_3 are extensions of M_1.



(a) M_1 (b) M_2 (c) M_3

Fig. 2. Some mergings of P and Q

Fig. 3 and Fig. 4 show the lattices $\mathfrak{M}$ and $\mathfrak{M}^\bullet$ of all mergings and of all proper mergings for our example. Please note that the labels show the (up to four-element) quasiordered sets $(P \cup Q, \leq_{R,S})$ for all (proper) mergings (R, S). We thereby used white nodes for elements from P and black nodes for elements from Q. In this very simple example we are allowed to omit the labels 1, 2, a, or b for the elements of the quasiordered sets on $P \cup Q$. There are 29 mergings whereof 20 are proper.

3.3 The Non-disjoint Case

The aim of this subsection is to describe how the case of *non-disjoint* ordered sets can be embedded into the previously described theory. So let us now assume that P and Q are not necessarily disjoint. We define $A := P \cap Q$ to be the intersection. A necessary, but very reasonable assumption is that the restrictions of $\leq_P$ and $\leq_Q$ to A are equal, i.e., that for $a, b \in A$ it follows that

$$a \leq_P b \iff a \leq_Q b.$$

The self-evident way to reduce this setting to the disjoint case is to carry on to assume that P and Q are disjoint, but to additionaly choose a subset A of P

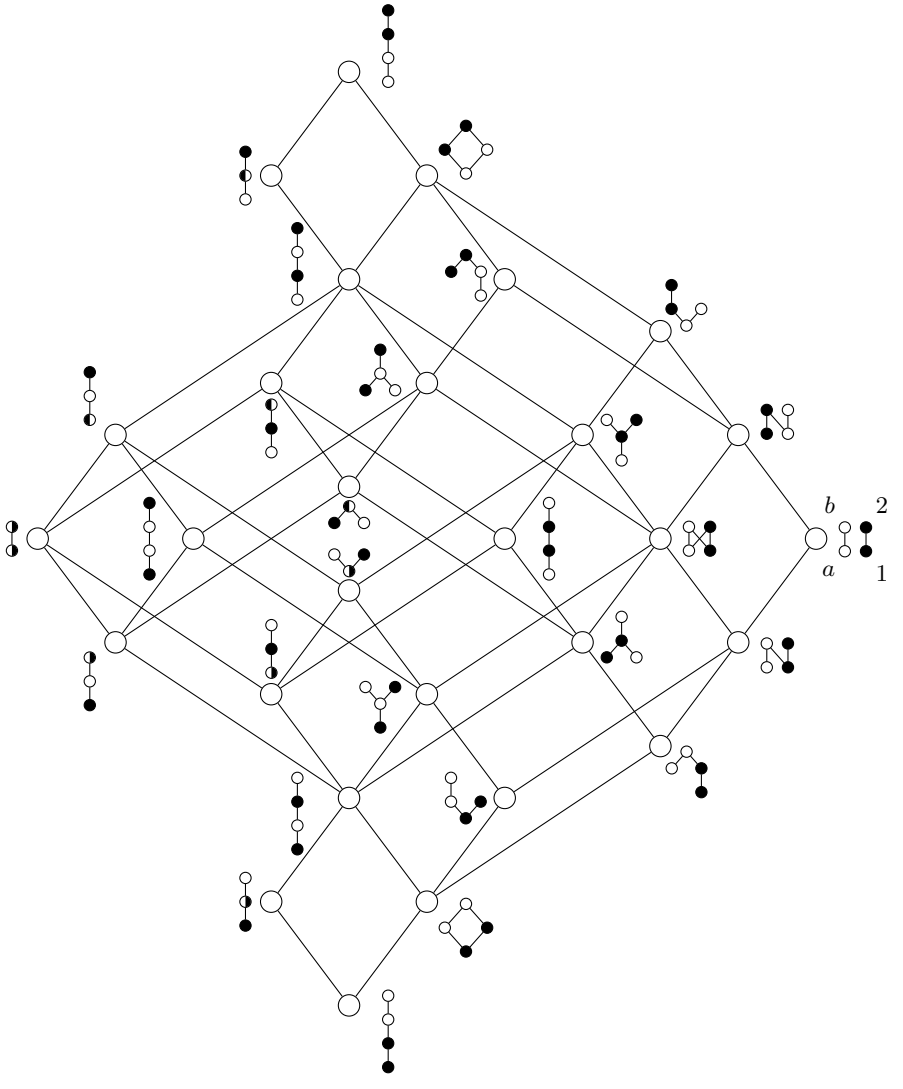


Fig. 3. The lattice $\mathfrak{M}$ of all mergings for the two two-element chains from Fig. 1. Thereby, the nodes that are half-filled correspond to the cases where elements from P and Q are equivalent in the merged quasiordered sets. Hence, the quasiordered sets without such a half-filled node correspond to the proper mergings. The complete sublattice $\mathfrak{M}^\bullet$ of all proper mergings is displayed in the following Fig. 4.

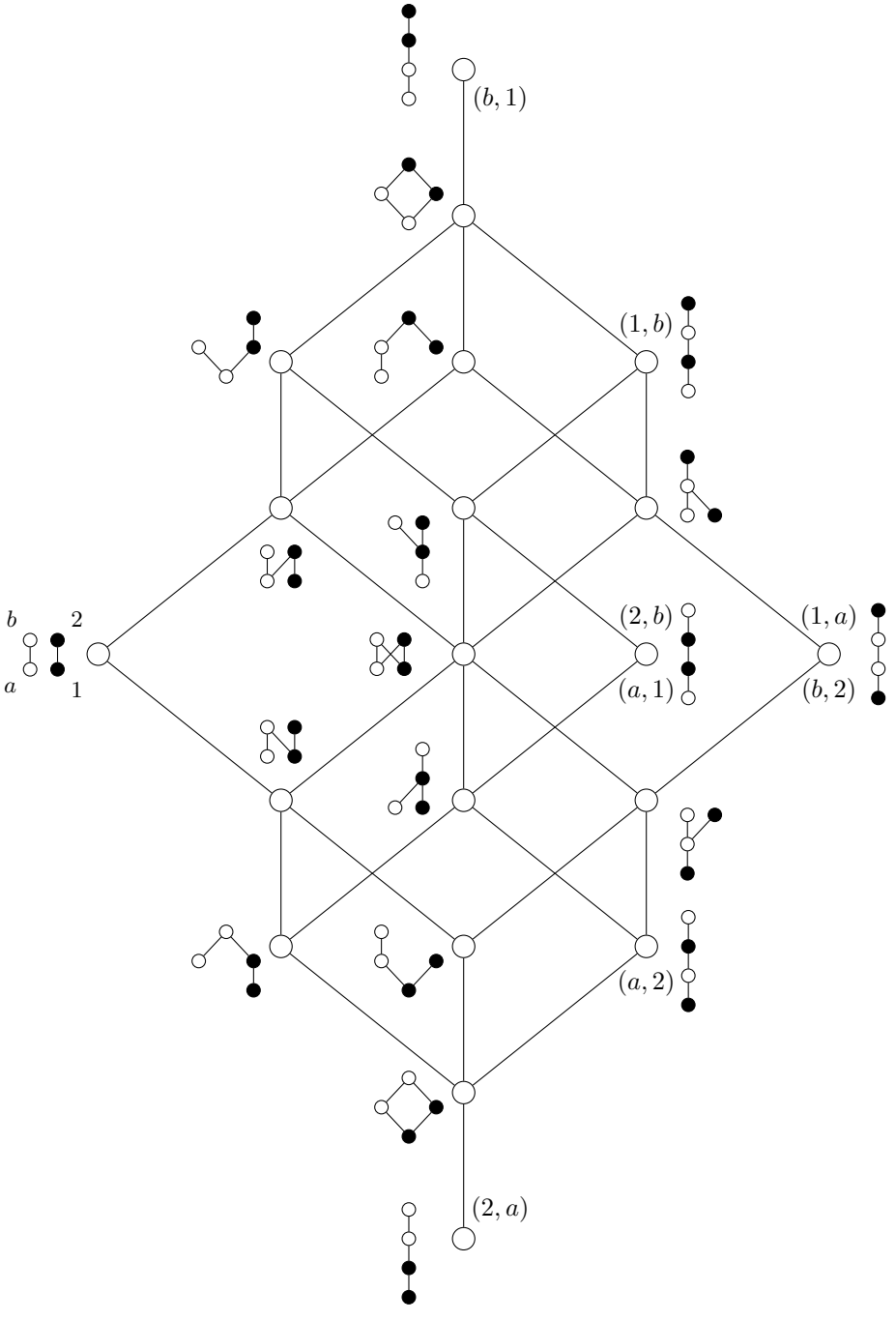


Fig. 4. The lattice $\mathfrak{M}^\bullet$ of proper mergings for the two two-element chains. In order to understand the labels like $(a, 1) \in P \times Q$ or $(1, b) \in Q \times P$ we refer the reader to Theorem 2 below. A representing formal context is displayed in Fig. 7.

and an order embedding $\psi : A \to Q$ modeling the *identification*. Hence, for all $a, b \in A$ we assume that

$$a \leq_P b \iff \psi a \leq_Q \psi b.$$

Then ψa can be understood as the copy of a in Q. Since we want a to be the copy of ψa in P, the embedding ψ should be one-to-one. In order to ensure this, we assume that P and Q are posets, i.e., that the respective order relations are antisymmetric. By Remark 1 this restriction does not lead to a lack of generality.

Definition 2. *Let P and Q be disjoint partially ordered sets, let $A \subseteq P$ and let $\psi : A \to Q$ be an order embedding, i.e., for $a, b \in A$ it follows that $a \leq b$ if and only if $\psi a \leq \psi b$ is satisfied. Then a merging (R, S) of P and Q is called a* **ψ-gluing** *if for every $a \in A$ the two conditions $(a, \psi a) \in R$ and $(\psi a, a) \in S$ are satisfied. A ψ-gluing (R, S) is said to be a* **proper ψ-gluing** *if for all $p \in P$ and $q \in Q$ the implication*

$$(p, q) \in R \text{ and } (q, p) \in S \implies p \in A \text{ and } q = \psi p.$$

is valid. The set of all (proper) ψ-gluings is denoted by $\mathfrak{M}_\psi$ (by $\mathfrak{M}^\bullet_\psi$).

Hence, a merging (R, S) of the two given posets P and Q is a ψ-gluing if for every $a \in A$ it follows that

$$a \leq_{R,S} \psi a \leq_{R,S} a.$$

This means that every element from A joins its copy in Q to form the only non-singleton equivalence classes in the factor poset of $(P \cup Q, \leq_{R,S})$. Note that if one chooses A to be the empty set and ψ to be the only mapping from $\emptyset$ to Q, the definition from above yields mergings and proper mergings again.

Proposition 3. *Let P and Q be disjoint posets, let $A \subseteq P$ and let $\psi : A \to Q$ be an order-embedding. Then a merging (R, S) of P and Q is a ψ-gluing iff $R_0 \subseteq R$ and $S_0 \subseteq S$, where*

$$p \, R_0 \, q \iff \exists a \in A : \, p \leq a \text{ and } \psi a \leq q,$$

$$q \, S_0 \, p \iff \exists a \in A : \, q \leq \psi a \text{ and } a \leq p.$$

The set $\mathfrak{M}_\psi$ of all ψ-gluings forms an interval in $\mathfrak{M}$.

Proof. It is not hard to see that a merging (R, S) is a ψ-gluing iff $R_0 \subseteq R$ and $S_0 \subseteq S$. We show that (R_0, S_0) is a merging. Because then the rest follows from Corollary 2. One can see from the definition that for every $p \in P$ the row p^{R_0} is the union of order filters in Q, and hence is an order filter itself:

$$p^{R_0} = \bigcup \{ \uparrow_Q \psi a \mid a \in A \cap \uparrow_P a \}.$$

Analogously, one shows that for every $q \in Q$ the column q^{R_0} is an order ideal in P, which by Proposition 1 makes R_0 an order ideal in $P \times Q^d$. In the same way

one can show that S_0 is an order filter in $Q^d \times P$. Furthermore, for $p_1, p_2 \in P$ it follows that

$$p_1 \, R_0 \circ S_0 \, p_2 \iff \exists q \in Q \, \exists a_1, a_2 \in A : \; p_1 \leq a_1 \text{ and } \psi a_1 \leq q \leq \psi a_2 \text{ and } a_2 \leq p_2$$
$$\implies \exists a_1, a_2 \in A : \; p_1 \leq a_1 \leq a_2 \leq p_2$$

Hence, it follows that $R_0 \circ S_0 \subseteq \leq_P$. Note that for the implication we needed that ψ is an order embedding. Dually, one can show that $S_0 \circ R_0 \subseteq \leq_Q$. By Proposition 2 the pair (R_0, S_0) is a merging. □

4 Contextual Representation

In the following we introduce formal contexts representing the lattices of mergings and lattices of proper mergings. Furthermore, we discuss related topics like how one can describe the intervals $\mathfrak{M}_{X,Y}$ from Corollary 2 by formal contexts, or how to read the maximal mergings from the representing context.

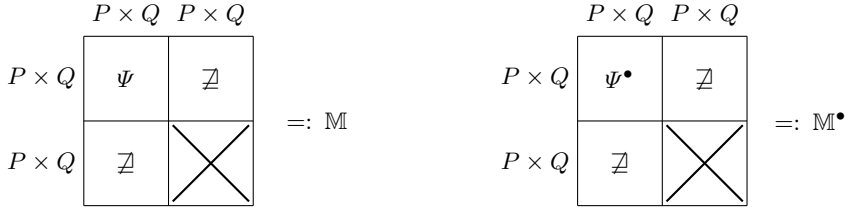


Fig. 5. The representing contexts $\mathbb{M}$ and $\mathbb{M}^\bullet$ of the lattices $\mathfrak{M}$ and $\mathfrak{M}^\bullet$ of all mergings on the left and of all proper mergings on the right. Thereby, $\sqsubseteq$ denotes the order on $P \times Q^d$. Hence, we have that $(p_1, q_1) \sqsubseteq (p_2, q_2)$ iff $p_1 \leq p_2$ and $q_1 \geq q_2$. For the definition of the relations Ψ and $\Psi^\bullet$ in the upper left quadrants we refer to Theorem 2 below. Please note that the big cross in the lower right quadrant represents the *universal relation*, i.e., every pair from $P \times Q$ is in relation to every pair from $P \times Q$. To be precise, the object (and attribute) set of the displayed formal contexts is the disjoint union $(P \times Q) \uplus (P \times Q)$, where for two sets A and B one puts $A \uplus B := (\{1\} \times A) \cup (\{2\} \times B)$. And in order to be technically correct one thinks of the displayed contexts $\mathbb{M}$ and $\mathbb{M}^\bullet$ as the composition of the four respective disjoint copies of the displayed non-disjoint parts.

Theorem 2. *Let P and Q be disjoint quasiordered sets. Then the following statements hold:*

(i) The lattice $\mathfrak{M}$ of all mergings of P and Q is isomorphic to the concept lattice of the context $\mathbb{M}$ displayed in Fig. 5. Thereby the relation Ψ from the upper left quadrant is given by

$$(p_1, q_1) \, \Psi \, (p_2, q_2) \; :\iff \begin{cases} q_1 \leq q_2 \Rightarrow p_1 \leq p_2, \text{ and} \\ p_1 \geq p_2 \Rightarrow q_1 \geq q_2. \end{cases}$$

An isomorphism $\varphi : \mathfrak{M} \to \underline{\mathfrak{B}}(\mathbb{M})$ *is given by*

$$(R, S) \longmapsto \left(R \uplus (S^{-1})^{\complement}, S^{-1} \uplus R^{\complement}\right).$$

(ii) The lattice $\mathfrak{M}^{\bullet}$ *of all proper mergings is isomorphic to the concept lattice of the context* $\mathbb{M}^{\bullet}$ *displayed in Fig. 5. Thereby the relation* $\Psi^{\bullet}$ *from the upper left quadrant is given by*

$$(p_1, q_1)\, \Psi^{\bullet}\, (p_2, q_2) \;:\Longleftrightarrow\; \begin{cases} q_1 \leq q_2 \Rightarrow p_1 < p_2, & \text{and} \\ p_1 \geq p_2 \Rightarrow q_1 > q_2 \end{cases}$$

An isomorphism $\varphi : \mathfrak{M}^{\bullet} \to \underline{\mathfrak{B}}(\mathbb{M}^{\bullet})$ *is given by*

$$(R, S) \longmapsto \left(R \uplus (S^{-1})^{\complement}, S^{-1} \uplus R^{\complement}\right).$$

Proof. We have seen in Theorem 1 that the lattice of mergings and the lattice of proper mergings are both complete sublattices of

$$\mathcal{OI}(P \times Q^d) \times \left(\mathcal{OF}(Q^d \times P)\right)^d.$$

Since the mapping that maps an order filter to its complement is an anti-automorphism and since the inversion $(\cdot)^{-1}$ easily transforms order filters from $P \times Q^d$ into those of $Q^d \times P$ (and vice versa), we get that the following mapping

$$\begin{aligned} \hat{\varphi} : \mathcal{OI}(P \times Q^d) \times \left(\mathcal{OF}(Q^d \times P)\right)^d &\longrightarrow \underline{\mathfrak{B}}(\mathbb{S}) \\ (R, S) &\longmapsto \left(R \uplus (S^{-1})^{\complement}, S^{-1} \uplus R^{\complement}\right). \end{aligned}$$

is an isomorphism. Thereby $\mathbb{S}$ denotes the direct sum of the contraordinal scale $(P \times Q, P \times Q, \not\ni)$ with itself. This direct sum is displayed in Fig. 6. Theorem 13 in [GW99] describes a one-to-one correspondence between the complete sublattices of a concept lattice and the so-called closed subrelations of the underlying formal context. What we will do in the following is to describe the two closed subrelations for our two complete sublattices.

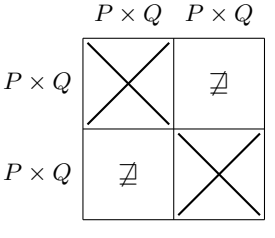


Fig. 6. The direct sum $\mathbb{S} = (P \times Q, P \times Q, \not\ni) + (P \times Q, P \times Q, \not\ni)$ of two contraordinal scales; see [GW99] Definition 32

For the upper right quadrant we obtain the following relation

$$\bigcup \{R \times R^{\complement} \mid (R, S) \text{ merging}\}$$

which by Proposition 1 equals the relation $\not\supseteq$. Please note that in this case we have $R^{\complement} = R^{\not\supseteq}$. Analogously one obtains that the lower left part of the closed subrelation belonging to $\mathfrak{M}$ is also $\not\supseteq$. For the lower right quadrant one gets

$$\bigcup\{(S^{-1})^{\complement} \times R^{\complement} \mid (R,S) \text{ merging}\}$$

which is $(P \times Q)^2$ since $(\emptyset,\emptyset)$ is a merging. We denote the remaining upper left part of the closed subrelation belonging to $\mathfrak{M}$ by Ψ. In order to determine Ψ we make some preliminary thoughts. Let $p_1, p_2 \in P$ and let $q_1, q_2 \in Q$. Then it follows that

$$\big(\downarrow p_1 \times \uparrow q_1\big) \circ \big(\downarrow q_2 \times \uparrow p_2\big) \;=\; \begin{cases} \emptyset & \text{if } q_1 \not\leq q_2, \\ \big(\downarrow p_1 \times \uparrow p_2\big) & \text{else.} \end{cases}$$

Hence, it follows that

$$\big(\downarrow p_1 \times \uparrow q_1\big) \circ \big(\downarrow q_2 \times \uparrow p_2\big) \subseteq \leq_P \iff \big(q_1 \leq q_2 \Rightarrow p_1 \leq p_2\big).$$

Dually one gets

$$\big(\downarrow q_2 \times \uparrow p_2\big) \circ \big(\downarrow p_1 \times \uparrow q_1\big) \subseteq \leq_Q \iff \big(p_2 \leq p_1 \Rightarrow q_2 \leq q_1\big).$$

From this we conclude (with the help of Proposition 2) that

$$\begin{aligned}
(p_1,q_1)\,\Psi\,(p_2,q_2) &\iff \exists (R,S) \in \mathfrak{M} :\ p_1\ R\ q_1 \text{ and } q_2\ S\ p_2 \\
&\iff \exists R \in \mathcal{OI}(P \times Q^d), \exists S \in \mathcal{OF}(Q^d \times P) : \\
&\qquad R \circ S \subseteq \leq_P,\ S \circ R \subseteq \leq_Q,\ (p_1,q_1) \in R \text{ and } (q_2,p_2) \in S \\
&\iff \text{for } R := \downarrow p_1 \times \uparrow q_1 \text{ and } S := \downarrow q_2 \times \uparrow p_2 \text{ it follows that} \\
&\qquad R \circ S \subseteq \leq_P \text{ and } S \circ R \subseteq \leq_Q \\
&\iff \big(p_2 \leq p_1 \Rightarrow q_2 \leq q_1\big) \text{ and } \big(q_1 \leq q_2 \Rightarrow p_1 \leq p_2\big).
\end{aligned}$$

The determination of the closed subrelation belonging to the complete sublattice $\mathfrak{M}^{\bullet}$ of proper mergings can be done analogously. The only difference occurs in the upper left part which will be denoted by $\Psi^{\bullet}$. One obtains

$$\begin{aligned}
(p_1,q_1)\,\Psi^{\bullet}\,(p_2,q_2) &\iff \exists (R,S) \in \mathfrak{M}^{\bullet} :\ p_1\ R\ q_1 \text{ and } q_2\ S\ p_2 \\
&\iff \text{for } R := \downarrow p_1 \times \uparrow q_1 \text{ and } S := \downarrow q_2 \times \uparrow p_2 \text{ it follows that} \\
&\qquad R \circ S \subseteq \leq_P \text{ and } S \circ R \subseteq \leq_Q \text{ and } R \cap S^{-1} = \emptyset \\
&\iff \big(p_2 \leq p_1 \Rightarrow q_2 \leq q_1\big) \text{ and } \big(q_1 \leq q_2 \Rightarrow p_1 \leq p_2\big) \text{ and} \\
&\qquad \big(p_2 \not\leq p_1 \text{ or } q_1 \not\leq q_2\big) \\
&\iff \big(p_2 \leq p_1 \Rightarrow q_2 < q_1\big) \text{ and } \big(q_1 \leq q_2 \Rightarrow p_1 < p_2\big).
\end{aligned}$$

□

Remark 2. One can easily show that Ψ and $\Psi^\bullet$ are self-bonds of the contraordinal scale $(P \times Q, P \times Q, \not\sqsupseteq)$. Furthermore, it follows that $\Psi^\bullet \subseteq \Psi$. If both, P and Q are chains, it follows that $\not\sqsupseteq = \Psi^\bullet$.

Example 1. The previous theorem describes how one can construct a formal context that represents the lattice of (proper) mergings. Fig. 7 shows the formal context, whose concept lattice is depicted in Fig. 4. As we notice, the concept lattice of $\mathbb{M}^\bullet$ in Fig. 4 is not labeled with the usual reduced labeling for concept lattices. Since the mergings can be understood as certain preconcepts of the upper left quadrant, we just use the objects and attributes belonging to this quadrant to label the lattice. Furthermore, as attribute labels we use the pairs from $Q \times P$ instead of the corresponding inverse pairs pairs from $P \times Q$. This makes it more intuitive to read a (proper) merging (R, S) from the diagram. Thereby, R contains all pairs in $P \times Q$ that are object labels of concepts below and S contains all pairs in $Q \times P$ that are attribute labels of concepts above this concept.

	$a1$	$a2$	$b1$	$b2$	$a1$	$a2$	$b1$	$b2$
$a1$			×	×			×	×
$a2$	×		×	×	×		×	×
$b1$								
$b2$	×		×		×		×	
$a1$			×	×	×	×	×	×
$a2$	×		×	×	×	×	×	×
$b1$					×	×	×	×
$b2$	×		×		×	×	×	×

Fig. 7. The context $\mathbb{M}^\bullet$

Let us, for example, have a look at the merging M_3 as given in Fig. 2(c). The object labels that are attached to concepts below the respective concept in $\mathfrak{M}^\bullet$ are $(b, 2)$ and $(a, 2)$. Thus, we have $R = \{(a, 2), (b, 2)\}$. The only attribute label attached to a concept above the corresponding concept is $(1, b)$ and thus $S = \{(1, b)\}$. This matches exactly the observation from Fig. 2(c).

Let (R, S) be a fixed merging. In Corollary 2 we have seen that the set $\mathfrak{M}_{R,S}$ of all extensions of (R, S) forms an interval in $\mathfrak{M}$. Furthermore, we learned in Proposition 3 that such a setting might for instance be used to describe the case were the two ordered sets P and Q are not disjoint. It is not hard to see from the previous Theorem 2 that for arbitrary relations $X \subseteq P \times Q$ and $Y \subseteq Q \times P$ the set $\mathfrak{M}_{X,Y}$ is nonempty if and only if (X, Y) is a preconcept of the upper left quadrant of $\mathbb{M}$, i.e., if $X \times Y \subseteq \Psi$. Analogously, the set $\mathfrak{M}^\bullet_{X,Y}$ is nonempty if and only if $X \times Y \subseteq \Psi^\bullet$. Hence, in these cases the intervals $\mathfrak{M}_{X,Y}$ and $\mathfrak{M}^\bullet_{X,Y}$ form complete lattices. The question that shall be answered in the following Proposition 4 is how contextual representations of these intervals look like.

Proposition 4. *Let P and Q be disjoint quasiordered sets and let*[2] *$X, Y \subseteq P \times Q$. Furthermore, let $\downarrow X$ denote the smallest order ideal in $P \times Q^d$ containing X and let $\uparrow Y$ denote the smallest order filter in $P \times Q^d$ containing Y.*

[2] Instead of $Y \subseteq Q \times P$ we choose a relation $Y^{-1} \subseteq Q \times P$. This replacement of Y by Y^{-1} makes the description of the representing contexts easier.

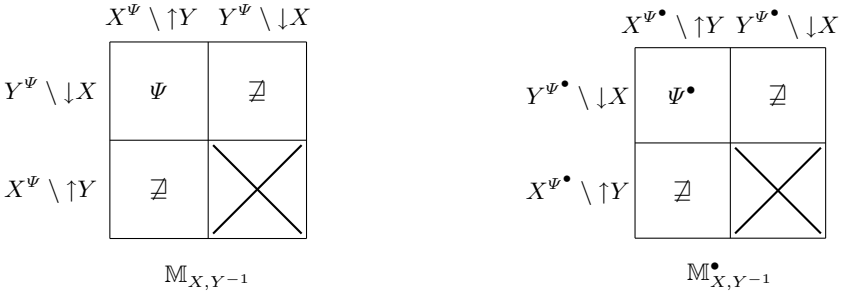


Fig. 8. The representing contexts $\mathbb{M}_{X,Y^{-1}}$ and $\mathbb{M}^\bullet_{X,Y^{-1}}$ of the complete lattices $\mathfrak{M}_{X,Y^{-1}}$ on the left and $\mathfrak{M}^\bullet_{X,Y^{-1}}$ on the right. Thereby, $\sqsubseteq$, Ψ, $\Psi^\bullet$ and the big cross denote the restrictions of the respective relations from Fig. 5. Hence, the two contexts are subcontexts of $\mathbb{M}$ and $\mathbb{M}^\bullet$, respectively.

(i) If $X \times Y \subseteq \Psi$, then the interval $\mathfrak{M}_{X,Y^{-1}}$ is isomorphic to the concept lattice of the context $\mathbb{M}_{X,Y^{-1}}$ displayed in Fig. 8. An isomorphism is given by

$$\begin{aligned} \xi : \quad \underline{\mathfrak{B}}(\mathbb{M}_{X,Y^{-1}}) &\longrightarrow \mathfrak{M}_{X,Y^{-1}} \\ (A_1 \uplus A_2, B_1 \uplus B_2) &\longmapsto ({\downarrow}X \cup A_1, ({\uparrow}Y \cup B_1)^{-1}). \end{aligned}$$

(ii) If $X \times Y \subseteq \Psi^\bullet$, then the interval $\mathfrak{M}^\bullet_{X,Y^{-1}}$ is isomorphic to the concept lattice of the context $\mathbb{M}^\bullet_{X,Y^{-1}}$ displayed in Fig. 8. An isomorphism is given by

$$\begin{aligned} \xi : \quad \underline{\mathfrak{B}}(\mathbb{M}^\bullet_{X,Y^{-1}}) &\longrightarrow \mathfrak{M}^\bullet_{X,Y^{-1}} \\ (A_1 \uplus A_2, B_1 \uplus B_2) &\longmapsto ({\downarrow}X \cup A_1, ({\uparrow}Y \cup B_1)^{-1}). \end{aligned}$$

Thereby, for a concept (A, B) of $\mathbb{M}_{X,Y^{-1}}$ or of $\mathbb{M}^\bullet_{X,Y^{-1}}$ the disjoint unions $A = A_1 \uplus A_2$ and $B = B_1 \uplus B_2$ describe the split-up into the object and attribute sets of the four quadrants. Thus, one reads the mergings from the upper left quadrants.

Proof. One can easily show that for every quasiordered set $(Z, \leq)$ and every $W \subseteq Z$ the context $(W, W, \not\geq)$ is a compatible[3] subcontext of the whole contraordinal scale $(Z, Z, \not\geq)$. It is now an easy application of [GW99, Proposition 35] to show that the formal context displayed in Fig. 9 is a compatible subcontext of the direct sum displayed in Fig. 6. With the help of [GW99, Proposition 48] and the proof of Theorem 2 we infer that $\mathbb{M}_{X,Y^{-1}}$ from Fig. 8 is a compatible subcontext of $\mathbb{M}$.

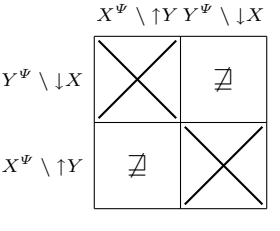


Fig. 9. A compatible subcontext of the direct sum from Fig. 6

[3] See [GW99, Definition 45].

We put $\dot{X} := Y^{\Psi} \setminus {\downarrow}X$ and $\dot{Y} := X^{\Psi} \setminus {\uparrow}Y$. By [GW99, Proposition 34] the mapping $\mathit{\Pi} : \underline{\mathfrak{B}}(\mathbb{M}) \to \underline{\mathfrak{B}}(\mathbb{M}_{X,Y^{-1}})$ that restricts the extents and intents of concepts from $\mathbb{M}$ to the object set $\dot{X} \uplus \dot{Y}$ and to the attribute set $\dot{Y} \uplus \dot{X}$ of $\mathbb{M}_{X,Y^{-1}}$ is a surjective complete homomorphism. In particular it is a mapping. Furthermore, $\iota : \mathfrak{M}_{X,Y^{-1}} \to \mathfrak{M}$ with $(R, S) \mapsto (R, S)$ is an order embedding and φ from Theorem 2 (i) is an isomorphism from $\mathfrak{M}$ to $\underline{\mathfrak{B}}(\mathbb{M})$. We put

$$\xi^{-1} := \mathit{\Pi} \circ \varphi \circ \iota$$

and show that it is an isomorphism from $\mathfrak{M}_{X,Y^{-1}}$ to $\underline{\mathfrak{B}}(\mathbb{M}_{X,Y^{-1}})$. It is then easy to see that its inverse is the mapping ξ described above in (i).

For $(R, S) \in \mathfrak{M}_{X,Y^{-1}}$ it follows that

$$\begin{aligned}\xi^{-1}(R, S) &= \mathit{\Pi}\big(\varphi(R, S)\big)\\ &= \big((R \cap \dot{X}) \uplus ((S^{-1})^{\complement} \cap \dot{Y}), (S^{-1} \cap \dot{Y}) \uplus (R^{\complement} \cap \dot{X})\big).\end{aligned}$$

From $R \subseteq (S^{-1})^{\Psi}$ and $S^{-1} \supseteq Y$ we infer that $R \subseteq Y^{\Psi}$ and hence $R \cap \dot{X} = R \setminus {\downarrow}X$. Analogously one shows that $S^{-1} \cap \dot{Y} = S^{-1} \setminus {\uparrow}Y$. With the help of this, we infer that the following implications are valid for $(R_i, S_i) \in \mathfrak{M}_{X,Y^{-1}}$ $(i = 1, 2)$:

$$\begin{aligned}\xi^{-1}(R_1, S_1) \leq \xi^{-1}(R_2, S_2) &\implies R_1 \setminus {\downarrow}X \subseteq R_2 \setminus {\downarrow}X \text{ and } S_1^{-1} \setminus {\uparrow}Y \supseteq S_2^{-1} \setminus {\uparrow}Y\\ &\iff R_1 \subseteq R_2 \text{ and } S_1 \supseteq S_2.\end{aligned}$$

Since ξ^{-1} is order-preserving for obvious reasons, we have shown that it is an order embedding and it remains to show that it is onto. Let therefore $(A_1 \uplus A_2, B_1 \uplus B_2)$ be a concept of $\mathfrak{M}_{X,Y^{-1}}$. Since the upper right and the lower left quadrant are compatible subcontexts, we infer that (A_1, B_2) and (A_2, B_1) are formal concepts of the respective quadrants. One can use this to show that for $R := A_1 \cup {\downarrow}X$ and $S := B_1 \cup {\uparrow}Y$ it follows that $\xi^{-1}(R, S) = (A_1 \uplus A_2, B_1 \uplus B_2)$. It remains to show that (R, S) is a merging. Since A_1 is an order ideal in $(\dot{X}, \sqsubseteq)$ we infer that R is an order ideal in $(Y^{\Psi}, \sqsubseteq)$. Since Y^{Ψ} is an order ideal in the whole $P \times Q^d$ (see Remark 2) we infer that R is an order ideal in $P \times Q^d$; see Fig. 10. Analogously, one can show that S^{-1} is an order filter in $P \times Q^d$. It

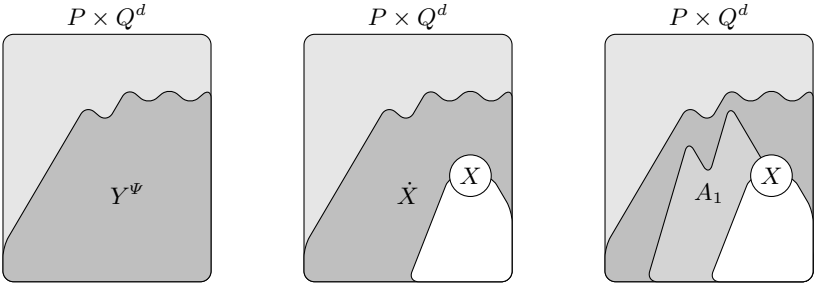


Fig. 10. Some order ideals in $P \times Q^d$

remains to show that $R \times S \subseteq \Psi$ (see the proof of Theorem 2). From $X \times Y \subseteq \Psi$ we infer with the help of Remark 2 that $\uparrow Y \subseteq (\downarrow X)^{\Psi}$. Furthermore, we infer from $A_1 \subseteq Y^{\Psi}$ and $\dot{Y}^{\complement} \supseteq Y$ that

$$A_1^{\Psi} = (A_1^{\Psi} \cap \dot{Y}) \cup (A_1^{\Psi} \cap \dot{Y}^{\complement}) \supseteq B_1 \cup (A_1^{\Psi} \cap Y) \supseteq B_1 \cup Y.$$

By Remark 2 this yields $A_1^{\Psi} \supseteq B_1 \cup \uparrow Y$. Dually, one can show $B_1^{\Psi} \supseteq A_1 \cup \downarrow X$, which implies $B_1 \subseteq (\downarrow X)^{\Psi}$. All this yields

$$S^{-1} = B_1 \cup \uparrow Y \subseteq A_1^{\Psi} \cap (\downarrow X)^{\Psi} = (A_1 \cup \downarrow X)^{\Psi} = R^{\Psi}.$$

Hence, (R, S) is indeed a merging and we have shown that ξ^{-1} is an isomorphism. Statement (ii) can be shown analogously. □

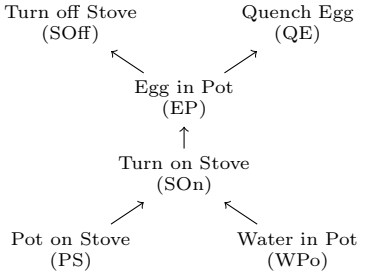
As an immediate consequence we get that a merging (R, S) is maximal if and only if (R, S^{-1}) is a formal concept of the upper left quadrant of $\mathbb{M}$. In this case all quadrants of $\mathbb{M}_{R,S}$ with exception of the lower right one are empty. Hence, $\mathbb{M}_{R,S}$ just has one formal concept which corresponds to the only extension of (R, S) which obviously is the merging itself. Analogously, the formal concepts of the upper left quadrant of $\mathbb{M}^{\bullet}$ correspond one-to-one to the maximal proper mergings.

5 An Illustrating Example

As we already mentioned in the introduction, a possible application of our notion of mergings can be found in the field of *scheduling*. Scheduling deals with assigning a set of processors (and possibly given resources) to a set of tasks, such that all tasks can be completed under certain (imposed) constraints; [BEP+01]. A common scheduling example is to create an execution plan for a certain (multi-part) action. There will be some tasks that can be freely executed (unconstrained) and some tasks that need to be completed before the execution of another task can be started. Executing (optimal) execution plans for two different actions on the same set of processors results in the problem of finding an (optimal) merging of both plans. With the construction given in the previous sections we obtain a handy tool to restrict the search space for this challenge.

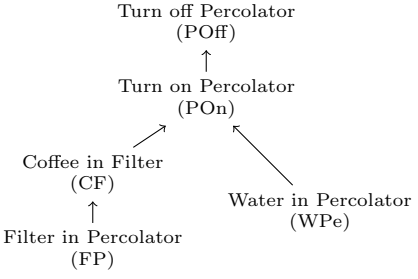
Taking the examples for scheduling plans as given in Fig. 11, we find that the lattice of all mergings between both posets has 521 138 elements and the sublattice of all proper mergings still has 403 844 elements. For practical use, this will be way too large. In order to find the optimal plan, a naïve approach would check all of these merged plans for optimality. However, if we assume a certain preordering for a desired merging, in the form of certain fixed relations in R and S, Proposition 4 allows for filtering suitable mergings in an intuitive manner. For our example we want to assume the following: Since boiling eggs lasts longer than brewing coffee, we do not want to start brewing coffee, until the water is boiling. Furthermore, we want to finish brewing coffee, before the eggs are done. Thus, we fix the following:

$$X := \{(\mathrm{EP}, \mathrm{FP}), (\mathrm{EP}, \mathrm{WPe})\}, \qquad Y := \{(\mathrm{POff}, \mathrm{SOff})\}$$



$\mathbb{E}$	PS	WPo	SOn	EP	SOff	QE
PS	×		×	×	×	×
WPo		×	×	×	×	×
SOn			×	×	×	×
EP				×	×	×
SOff					×	
QE						×

(a) Boiling an egg.



$\mathbb{C}$	FP	WPe	CF	POn	POff
FP	×		×	×	×
WPe		×		×	×
CF			×	×	×
POn				×	×
POff					×

(b) Brewing coffee.

Fig. 11. Two scheduling examples

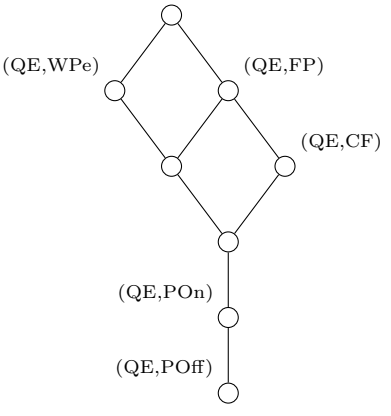


Fig. 12. The lattice $\mathfrak{M}^{\bullet}_{X,Y}$ of the proper mergings which extend (X, Y), where $X = \{(\mathrm{EP}, \mathrm{FP}), (\mathrm{EP}, \mathrm{WPe})\}$ and $Y = \{(\mathrm{POff}, \mathrm{SOff})\}$. Note the reduced labeling according to Theorem 2.

According to Corollary 2 the set of all mergings (R, S) with $X \subseteq R, Y \subseteq S$ forms an interval in $(\mathfrak{M}^{\bullet}, \leq)$. According to Proposition 4, our computation shows that we get only eight possible proper mergings, satisfying the constraints given by

X and Y. Fig. 12 shows the respective interval $\mathfrak{M}^{\bullet}_{X,Y}$ of $\mathfrak{M}^{\bullet}$. It shall be noted, that adding the constraint $(\mathrm{POff}, \mathrm{QE})$ to Y yields exactly one possible proper merging.

6 Conclusion and Outlook

The underlying article describes how to combine two given quasiordered sets without extending the initial orders. We started by classifying these mergings as special pairs (R, S) of order ideals R and order filters S^{-1} in a derived quasiordered set $P \times Q^d$. We then showed that these mergings form a complete lattice and described this lattice by a formal context. In the whole work we distinguished between the case of mergings and proper mergings, whereas the latter one can be understood as combinations of the initial orders where one does not allow to identify elements from the two different quasiordered sets.

To elaborate the topic in more detail, we described special intervals in the lattices of mergings belonging to extensions of a fixed pair of relations (X, Y) where $X \subseteq P \times Q$ and $Y \subseteq Q \times P$. We gave a contextual representation of these intervals as well, and concluded the paper with a possible application of these mergings in the field of scheduling, which was elaborated in an example.

The next, self-evident step to continue this work is to extend the notion of mergings towards mergings of an arbitrary number of ordered sets. Thereby, it is not hard to see how this generalisation can be done and how to characterise these mergings. But up to now we have not elaborated how the representing contexts might look like. We hope this generalistion could increase the readability and intuitivity of some of the statements and proofs presented in the underlying article.

It will turn out that this generalisation matches the application of Theorem 32 from [GW99] for the case of contraordinal scales. Hence, the mergings precisely describe subdirect products of distributive lattices. A way to describe subdirect products of lattices in FCA is a context construction called *P-product*. In this context a subsequent article should point out the connections to [Gan07, Gan08] and [KM06]. Namely, without explicitely mentioning it, we benefit from the well-known fact that there is a context representation of the Galois connections between distributive lattices. Furthermore, we want to mention that our lattices of mergings might be understood as lattices of approximations (as in [Mes10]) for special complete lattices.

References

[BEP+01] Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Scheduling Computer and Manufacturing Processes. Springer, Heidelberg (2001)

[DP02] Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press, Cambridge (2002)

[Gan07] Ganter, B.: Relational galois connections. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 1–17. Springer, Heidelberg (2007)

[Gan08] Ganter, B.: Lattices of Rough Set Abstractions as P-Products. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 199–216. Springer, Heidelberg (2008)

[GW99] Ganter, B., Wille, R.: Formal Concept Analysis: Mathematic Foundations. Springer, Heidelberg (1999)

[KM06] Krötzsch, M., Malik, G.: The tensor product as a lattice of regular galois connections. In: ICFCA 2006, pp. 89–104 (2006)

[Mes10] Meschke, C.: Approximations in concept lattices. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA 2010. LNCS, vol. 5986, pp. 104–123. Springer, Heidelberg (2010)

[Ste00] Steimann, F.: On the Representation of Roles in Object-Oriented and Conceptual Modelling. Data Knowledge Engineering 35(1), 83–106 (2000)

Mining Triadic Association Rules from Ternary Relations

Rokia Missaoui[1] and Léonard Kwuida[2]

[1] Université du Québec en Outaouais
rokia.missaoui@uqo.ca
[2] Zurich University of Applied Sciences - School of Engineering
Center for Applied Mathematics and Physics
CH-8401 Winterthur, Switzerland
kwuida@gmail.com

Abstract. Ternary and more generally n-ary relations are commonly found in real-life applications and data collections. In this paper, we define new notions and propose procedures to mine closed tri-sets (triadic concepts) and triadic association rules within the framework of triadic concept analysis. The input data is represented as a formal triadic context of the form $\mathbb{K} := (K_1, K_2, K_3, Y)$, where K_1, K_2 and K_3 are object, attribute and condition sets respectively, and Y is a ternary relation between the three sets. While dyadic association rules represent links between two groups of attributes (itemsets), triadic association rules can take at least three distinct forms. One of them is the following: $A \xrightarrow{C} D$, where A and D are subsets of K_2, and C is a subset of K_3. It states that A *implies* D *under the conditions in* C. In particular, the implication holds for any subset in C. The benefits of triadic association rules of this kind lie in the fact that they represent patterns in a more compact and meaningful way than association rules that can be extracted for example from the formal (dyadic) context

$$\mathbb{K}^{(1)} := (K_1, K_2 \times K_3, Y^{(1)}) \text{ with } (a_i, (a_j, a_k)) \in Y^{(1)} :\Longleftrightarrow\ (a_i, a_j, a_k) \in Y.$$

1 Introduction

The objective of this research work is to show how triadic association rules (including implications) can be generated from a triadic context together with triadic concepts (called also closed tri-sets). The present work can be useful to mine patterns from ternary relations between three groups of entity instances in general, and in particular when one of the three sets describes a collection of individuals while the other sets correspond to their properties (e.g., privileges or roles in secured systems) and the conditions (e.g., spatio-temporal constraints) under which they have such attributes.

In this section we briefly present work on closed set and implication computation from n-relations where $n \geq 3$. The theoretical basis for triadic concept analysis (TCA) has been defined by Wille and Lehmann [12,19] and generalized in [18] to n-adic formal contexts to produce n-adic formal concepts and complete n-lattices. Other studies closely related to TCA were conducted to generate patterns in the form of closed 3-sets (triadic concepts) [9] and triadic implications [3,5]. In [9], an algorithm called

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 204–218, 2011.

TRIAS allows the computation of triadic concepts from dyadic ones. Ji et al. [10] tackles the same problem of closed 3-set computation by proposing two algorithms: RSM and CubeMiner. The former relies on frequent closed 2-set mining to generate 3-sets while the latter exploits directly the tridimensional table to compute patterns in a more efficient manner by exploiting a ternary enumeration that recursively decomposes the dataset into smaller groups. The merit of the research presented in [4] lies in the generalization of the 3-set computation previously studied by [9,10] to a constraint-based mining approach for closed set computation from n-ary relations. The set of constraints contains piecewise (anti)-monotonic ones, including monotonic and antimonotonic constraints. An experimental comparison of DATA PEELER [4] with TRIAS and CubeMiner has been conducted and showed that the former behaves more efficiently than the two other algorithms for closed 3-set generation. The work in [14] defines a new semantics for inter-dimensional rules in dynamic oriented graphs that can be represented as n-ary relations with $n \geq 3$. In particular, a new objective interestingness measure called the exclusive confidence is proposed, and the computation of n-ary rules is allowed.

In this paper we propose novel notions and algorithms that use triadic concept analysis [12] as a framework to extract triadic concepts and generators as well as triadic association rules (including implications) of different types based on the studies in [3,5].

The remainder of this paper is organized as follows. In Section 2 we recall the basic definitions and notions related to formal concept analysis (FCA) and triadic concept analysis. Section 3 provides new definitions and notations. Then, we propose in Section 4 algorithms that generate triadic concepts, implications and more generally association rules by conducting a "factorization" on concepts, rules and generators extracted from a dyadic representation of the initial 3-dimensional one. Section 5 provides an experimental study while Section 6 concludes the paper and presents further work.

2 Triadic Concept Analysis

We will give some key definitions related to triadic concept analysis and illustrate them through the example in Figure 1. This is a triadic context borrowed from [5] but its meaning has been adapted to represent a data cube of three dimensions CUSTOMER, SUPPLIER, and PRODUCT. It concerns a group K_1 of customers (1 to 5) that purchase from suppliers in K_2 (**P**eter, **N**elson, **R**ick, **K**evin and **S**imon) products found in K_3 (**a**ccessories, **b**ooks, **c**omputers and **d**igital cameras). For example, the value ac at the cross of row 1 and column R means that customer 1 orders from supplier R products a and c. The right-hand side of Figure 1 is a dyadic context (extracted from the triadic one) where columns (attributes) are $(a_j, a_k) \in K_2 \times K_3$, usually denoted by $a_j \times a_k$ or simply a_j-a_k. We will often use simplified notations also for sets and tuples (e.g., 125 stands for $\{1, 2, 5\}$, ab for $\{a, b\}$, and P-aP-dN-dR-aK-a for $\{P\times a, P\times d, N\times d, R\times a, K\times a\}$).

2.1 Dyadic and Triadic Concepts

A formal (dyadic) context is a triple $\mathbb{K} := (G, M, I)$ where G, M and I stand for a set of objects, a set of attributes, and a binary relation between G and M respectively. For $A \subseteq G$ and $B \subseteq M$ two subsets $A' \subseteq M$ and $B' \subseteq G$ are defined as the set of

$\mathbb{K}$	P	N	R	K	S
1	abd	abd	ac	ab	a
2	ad	bcd	abd	ad	d
3	abd	d	ab	ab	a
4	abd	bd	ab	ab	d
5	ad	ad	abd	abc	a

$\mathbb{K}^{(1)}$	P				N				R				K				S			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
1	1	1		1	1	1		1	1		1		1	1			1			
2	1			1		1	1	1	1	1		1	1			1				1
3	1	1		1				1	1	1			1	1			1			
4	1	1		1		1		1	1	1			1	1						1
5	1			1	1			1	1	1		1	1	1	1		1			

Fig. 1. Left: A triadic context $\mathbb{K} := (K_1, K_2, K_3, Y)$, with $K_1 = \{1, 2, 3, 4, 5\}$ (customers), $K_2 = \{P, N, R, K, S\}$ (suppliers) and $K_3 = \{a, b, c, d\}$ (products). **Right**: The dyadic context $\mathbb{K}^{(1)}$ extracted from $\mathbb{K}$. Customers 1 to 5 purchase from suppliers **P**eter, **N**elson, **R**ick, **K**evin and **S**imon the products: **a**ccessories, **b**ooks, **c**omputers and **d**igital cameras.

attributes common to objects in A and the set of objects sharing all the attributes in B, respectively. Formally, the derivation $'$ is defined by

$$A' := \{a \in M \mid o\mathrm{I}a \ \forall o \in A\} \quad \text{and} \quad B' := \{o \in G \mid o\mathrm{I}a \ \forall a \in B\}.$$

This setting defines a pair of mappings $('\,,'\,)$ between the powerset of G and the powerset of M, which is a Galois connection. The induced closure operators (on G and M) are both denoted by $''$. For example, the closure of R-b is

$$\text{(R-b)}'' = (\text{(R-b)}')' = \{2, 3, 4, 5\}' = \{\text{P-a, P-d, N-d, R-a, R-b, K-a}\}.$$

A *formal concept* c is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A = B'$ and $B = A'$. The set A that we denote by $\mathrm{Ext}(c)$ is called the *extent* of c while B is its *intent* denoted by $\mathrm{Int}(c)$. A formal (dyadic) concept corresponds to a maximal rectangle (full of crosses / ones) in the dyadic context. In the closed *itemset* mining framework [16], G, M, A and B correspond to the transaction database, the set of items (products), the closed *tidset* and the closed *itemset* respectively.

The set $\mathfrak{B}(\mathbb{K})$ of all concepts of the context $\mathbb{K}$, partially ordered by: $(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2$ forms a complete lattice, called concept lattice of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$. A concept (X_2, Y_2) is called successor of a concept (X_1, Y_1) whenever $(X_1, Y_1) < (X_2, Y_2)$ holds. In this case, (X_1, Y_1) is called predecessor of (X_2, Y_2). The *immediate precedence* relation $\prec$ is the transitive reduction of $<$, i.e. $c_i \prec c_j$ if $c_i < c_j$ and there is no concept between c_i and c_j. We then call c_i an *immediate predecessor* of c_j and c_j an *immediate successor* of c_i. Figure 2 shows the Hasse diagram of the concept lattice corresponding to the dyadic context in Figure 1. The labeling of the diagram is reduced so that the extent of a concept represented by a node n is given by all labels in G (in white square) from the node n downwards, and the intent by all labels in M (in grey rectangles) from n upwards. For example, node #4 (on the right-hand side of the diagram) with the label **N-b** represents the concept $(\{1, 2, 4\}, \{\text{P-d, P-a, N-d, R-a, K-a, }\mathbf{\text{N-b}}\})$. The bottom of the lattice exhibits three attributes: S-b, S-c and P-c which indicate that no customer asks for books or computers from supplier S or for computers from supplier P.

Triadic concept analysis was originally introduced by Lehmann and Wille [12,19] as an extension to formal concept analysis, to analyze data described by three sets

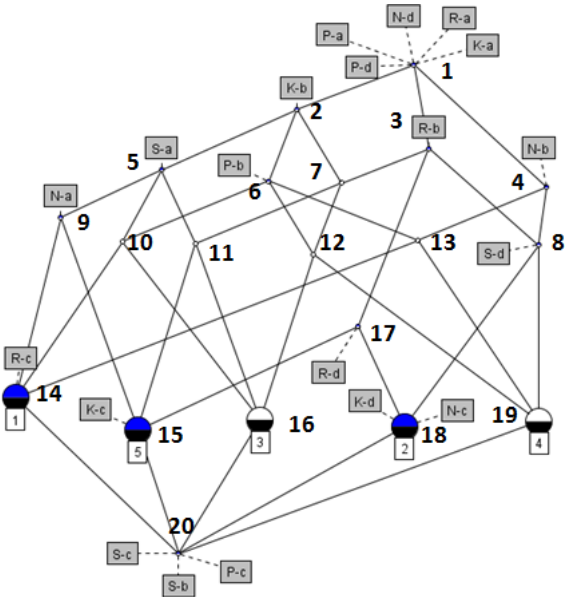


Fig. 2. Concept lattice generated from the dyadic context $\mathbb{K}^{(1)} := (K_1, K_2 \times K_3, Y^{(1)})$ where $(a_i, (a_j, a_k)) \in Y^{(1)} :\Leftrightarrow (a_i, a_j, a_k) \in Y$ (see the right-hand side of Figure 1)

K_1 (objects), K_2 (attributes) and K_3 (conditions) together with a 3-ary relation $Y \subseteq K_1 \times K_2 \times K_3$. $\mathbb{K} := (K_1, K_2, K_3, Y)$ is called a *triadic context*. A triple (a_1, a_2, a_3) in Y means that object a_1 possesses attribute a_2 under condition a_3. For e.g., the table on the left of Figure 1 is a triadic context (K_1, K_2, K_3, Y) representing the purchase of customers in $K_1 = \{1, 2, 3, 4, 5\}$ from suppliers in $K_2 = \{P, N, R, K, S\}$ of products in $K_3 = \{a, b, c, d\}$.

A *triadic concept* (also called *closed tri-set* or 3-*set* for short) of a triadic context is a triple (A_1, A_2, A_3) with $A_1 \subseteq K_1$, $A_2 \subseteq K_2$, $A_3 \subseteq K_3$ and $A_1 \times A_2 \times A_3 \subseteq Y$. It represents a maximal cuboid full with ones (or crosses). The subsets A_1, A_2 and A_3 are called the *extent*, the *intent* and the *modus* of the triadic concept (A_1, A_2, A_3) respectively. From Figure 1, we can extract e.g., the closed tri-sets $(12345, PRK, a)$ and $(14, PN, bd)$. The tri-set $(135, PN, d)$ is not closed since its extent can be augmented without violating the ternary relation to get $(12345, PN, d)$.

Let $\mathbb{K} := (K_1, K_2, K_3, Y)$ be a triadic context and $\{i, j, k\} = \{1, 2, 3\}$ with $j < k$. For $X_i \subseteq K_i$ and $(X_j, X_k) \subseteq K_j \times K_k$[1], an $^{(i)}$-derivation extending the derivation $'$ (see Subsection 2.1) is defined as follows [12]:

$$X_i^{(i)} := \{(a_j, a_k) \in K_j \times K_k \mid (a_i, a_j, a_k) \in Y \ \forall a_i \in X_i\}.$$

$$(X_j, X_k)^{(i)} := \{a_i \in K_i \mid (a_i, a_j, a_k) \in Y \text{ for all } (a_j, a_k) \in X_j \times X_k\}.$$

For example the $^{(1)}$-derivation in a triadic context $\mathbb{K} := (K_1, K_2, K_3, Y)$ is the derivation in the dyadic context $\mathbb{K}^{(1)} := (K_1, K_2 \times K_3, Y^{(1)})$ with $(a_i, (a_j, a_k)) \in Y^{(1)}$:

[1] We write $(X_j, X_k) \subseteq K_j \times K_k$ to mean that $X_j \subseteq K_j$ and $X_k \subseteq K_k$.

$\iff (a_i, a_j, a_k) \in Y$. In practice, the attribute×condition set can be restricted to the existing combinations (a_j, a_k) instead of all possible ones.

The set of triadic concepts can be ordered and form a complete trilattice [3,12]. Indeed, for each $i \in \{1, 2, 3\}$, the relation $(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3) \Leftrightarrow A_i \subseteq B_i$ is a quasi-order whose equivalence relation $\sim_i$ is given by: $(A_1, A_2, A_3) \sim_i (B_1, B_2, B_3) \Leftrightarrow A_i = B_i$. These three quasi-orders satisfy the following *antiordinal dependencies*: for $\{i, j, k\} = \{1, 2, 3\}$, $(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3)$ and $(A_1, A_2, A_3) \lesssim_j (B_1, B_2, B_3)$ imply $(B_1, B_2, B_3) \lesssim_k (A_1, A_2, A_3)$ for all concepts (A_1, A_2, A_3) and (B_1, B_2, B_3).

2.2 Dyadic and Triadic Association Rules

Let (G, M, I) be a formal dyadic context. An association rule is of the form $r : B \to C$ (s, c) where $B, C \subseteq M$ (*itemsets*) with $B \cap C = \emptyset$. The parameter $\text{s} = \text{supp}(r) = \frac{|B' \cap C'|}{|G|}$ is called the support of the rule r while $\text{c} = \text{conf}(r) = \frac{|B' \cap C'|}{|B'|}$ is its confidence [1]. An implication is an association rule whose confidence is equal to 1.

A set of studies in FCA were conducted on the generation of concise representations of rules [8,11] such as informative rules, Guigues-Duquenne base (stem base) [6,7], generic base [16], and Luxenburger base [13]. The notions of *generator* [16,17] and *pseudo-intent* [6,7] play a key role in such studies. A minimal generator of a closed itemset C is a subset B of C that is minimal w.r.t. $B'' = C$. A *generic basis* [16] associated with a given context is a concise representation of implications $B \to B'' \backslash B$ such that B is a minimal generator for B''. An *informative basis for approximate association rules* takes the following form: $B \to \text{Int}(c_i) \backslash B''$ where B is a minimal generator of $\text{Int}(c)$ and c_i is an immediate predecessor of c. The support of r is equal to $|\text{Ext}(c_i)|$ while its confidence is equal to $|\text{Ext}(c_i)|/|\text{Ext}(c)|$. For example, there are four (dyadic) generators for the intent of the concept #14 (see the lower left part of Figure 3). The generator R-c will produce the implication R-c → P-aP-bP-dN-aN-bN-dR-aK-aK-bS-a $(0.2, 1)$ while the generator P-bN-b associated with node #13 will generate the association rule P-bN-b → R-bS-d $(0.2, 0.5)$ when node #19 is considered.

To the best of our knowledge, Biedermann [3] was the first researcher who investigated the problem of implication extraction in triadic contexts. A *triadic implication* has the form $(A \to D)_C$ and holds if "*whenever A occurs under all conditions in C, then D also occurs under the same conditions*". Later on, Ganter and Obiedkov [5] extended Biedermann's work and defined three types of implications: *attribute×condition* implications (**AxCI**s), *conditional attribute* implications (**CAI**s), and *attributional condition* implications (**ACI**s).

An *attribute×condition implication* has the form $A \to D$, where A and D are subsets of $K_2 \times K_3$. Such implications (rather dyadic) are extracted from the binary context $\mathbb{K}^{(1)}$. For example, the implication R-c → P-aP-bP-dN-aN-bN-dR-aK-aK-bS-a $(0.2, 1)$ is an AxCI extracted from node #14 in Figure 3.

A *conditional attribute* implication takes the form: $A \xrightarrow{C} D$, where A and D are subsets of K_2, and C is a subset of K_3. It means that *A implies D under all conditions in C* and in particular, f*or any subset in C*. Such implication is then linked to Biedermann's definition of triadic implication as follows [5]: $A \xrightarrow{C} D \iff (A \to D)_{C_1}$ for all $C_1 \subseteq C$. As an illustration, $N \xrightarrow{ad} P$ holds since $(N \to P)_{C_1}$ is true for any $C_1 \subseteq \{a, d\}$.

Although $(N \rightarrow P)_{abd}$ holds, $N \xrightarrow{abd} P$ is not true since $(N \rightarrow P)_{C_1}$ does not hold for any $C_1 \subseteq \{a, b, d\}$ and in particular is not true for $C_1 \in \{b, bd\}$.

In a dual way, an *attributional condition* implication is an exact association rule of the form $A \xrightarrow{C} D$, where A and D are subsets of K_3, and C is a subset of K_2. Using our example, the CAI $N \xrightarrow{ad} P$ states that whenever **N**elson supplies **a**ccessories and **d**igital cameras (or any one of these two products), then **P**eter does so. The ACI: $b \xrightarrow{PN} d$ holds since whenever **b**ooks are supplied by both **P**eter and **N**elson, then **d**igital cameras are also provided by all these two suppliers. These last two kinds of implications (i.e., CAIs and ACIs) can be visualized through a concept lattice [5]. If we take the case of CAIs, a context (G, K_3, I) is first produced where G represents the set of implications of the form $A \rightarrow D$ such that A and D belong to K_2. The binary relation I holds between the implication $A \rightarrow D$ and a condition c in K_3 iff $A \xrightarrow{c} D$ holds. Based on the definitions given earlier, the formal concepts generated from such a context are couples (F, C) where F is a set of CAIs: $A \xrightarrow{C} D$ and C is the largest set of conditions for which a given implication holds. An example of CAI visualization is provided in Figure 4.

To distinguish triadic association rules (and implications) with Biedermann's meaning from their extensions defined by Ganter and Obiedkov, we will use the prefix **B** for the first group of patterns.

3 Data Mining from Triadic Contexts

The intuition behind our approach is the following: the extraction of *triadic* concepts, generators and association rules (including implications) from a triadic context can be obtained by first (i) flattening the initial context into a dyadic one (e.g., $\mathbb{K}^{(1)}$), then (ii) getting *dyadic* concepts, generators, attribute×condition association rules (AxCARs), and finally (iii) proceeding to a *factorization* of the output to get *triadic* concepts, generators and association rules, mainly those of the form $(A \rightarrow D)_C$ *i.e.*, with Biedermann's meaning. Note that the mapping $\mathbb{K} \leftrightarrow \mathbb{K}^{(i)}$ is one to one and onto, and therefore does not lead to any loss of information.

3.1 Definitions

Based on the key notions presented earlier [12], the following expression can be defined, where $(U_2, U_3) \subseteq K_2 \times K_3$:

$$(U_2, U_3)^{(1)(1)} := \left\{ (A_2, A_3) \mid U_i \subseteq A_i \subseteq K_i \text{ and } \left((U_2, U_3)^{(1)}, A_2, A_3 \right) \text{ is a triadic concept} \right\}.$$

Definition 1. *(U_2, U_3) is a t-generator (triadic generator) of the couple (A_2, A_3) as a part of a triadic concept (A_1, A_2, A_3) iff $(A_2, A_3) \in ((U_2, U_3)^{(1)})^{(1)}$.*

From Figures 1 and 3, one can see that $(PN, b)^{(1)} = \{1, 4\}$ and $\{1, 4\}^{(1)} = \{(PNK, b), (PN, bd)\}$. Therefore, $((PN, b)^{(1)})^{(1)} = \{(PNK, b), (PN, bd)\}$. Therefore, the pair (PN, b) is a t-generator for the pairs (PNK, b) and (PN, bd) associated with the extent $\{1, 4\}$.

Let $\mathbb{K} := (K_1, K_2, K_3, Y)$ be a triadic context. We denote by $\Pi_{2.3}(Y)$ the projection of Y on $K_2 \times K_3$. To extract triadic concepts we first flatten the triadic context $\mathbb{K}$ to get a binary context from which we extract the dyadic concepts. Note that if (A_1, A_2, A_3) is a triadic concept of $\mathbb{K}$, then there is a dyadic concept (C, D) of $(K_2, K_3, \Pi_{2.3}(Y))$ such that $A_2 \subseteq C$ and $A_3 \subseteq D$. To get triadic concepts and t-generators from the set of dyadic concepts and generators we define partial maps as follows.

Proposition 1. *Let (A_1, B) be a dyadic concept of $\mathbb{K}^{(1)}$. Then (K_2, K_3, B) is a dyadic context. We set $B_2 := \pi_1(B)$ and $B_3 := \pi_2(B)$. A tuple (A_1, A_2, A_3) with $A_1 \subseteq K_1$, $A_2 \subseteq B_2$ and $A_3 \subseteq B_3$ is a triadic concept of $\mathbb{K}^{(1)}$ if and only if the following conditions hold: (i) (A_2, A_3) is a dyadic concept in the sub-context (K_2, K_3, B), and (ii) $(A_2, A_3)^{(1)} = A_1$.*

Proposition 2. *Let g be a (dyadic) generator in $\mathbb{K}^{(1)}$, (and hence a subset of $K_2 \times K_3$). Let $U_2 := \Pi_1(g)$ and $U_3 := \Pi_2(g)$. Then (U_2, U_3) is a t-generator if and only if $|U_2| \times |U_3| = |g|$.*

From the concept in node #6 (see Figure 2) whose extent is $\{1, 3, 4\}$ and intent is $B = \{P \times a, P \times b, P \times d, N \times d, R \times a, K \times a, K \times b\}$, four triples can be generated. However, only $(134, PK, ab)$ and $(134, P, abd)$ are triadic concepts while $(134, PRK, a)$ and $(134, PN, d)$ are not closed 3-sets since in the last two cases $(A_2, A_3)^{(1)}$ leads to $\{1, 2, 3, 4, 5\} \neq \{1, 3, 4\}$.

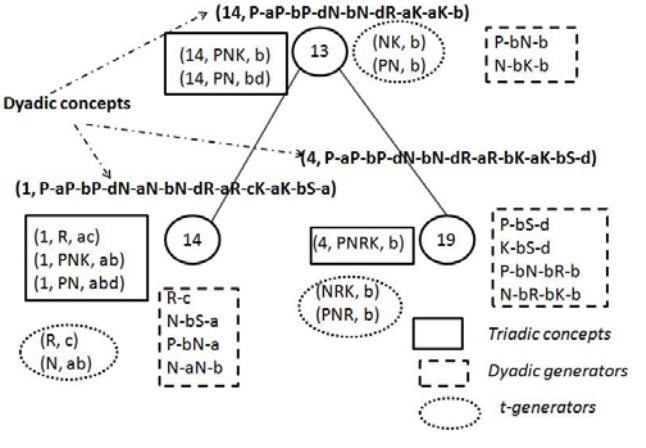


Fig. 3. Dyadic and triadic concepts and generators

A look at Figure 3 shows that there are two t-generators associated with the node #13 and their corresponding triadic concepts. Indeed, (NK, b) and (PN, b) are t-generators for the pairs (PNK, b) and (PN, bd) related to the extent $\{1, 4\}$. However, the dyadic generators N-bS-a (i.e., $g := \{N \times b, S \times a\}$) and P-bN-a extracted from the node #14 whose intent is P-aP-bP-dN-aN-bN-dR-aR-cK-aK-bS-a do not lead to t-generators since they do not verify the condition stated in Proposition 2.

3.2 *Attribute*×*condition* Association Rules

An *attribute*×*condition* association rule (AxCAR) is of the form: $A \longrightarrow D$ where each element in the sets A and D is of the form $a_j \times a_k$ such that $a_j \in K_2$ and $a_k \in K_3$. Using existing algorithms and theory (see Subsection 2.1), one can easily compute AxCARs from the context $\mathbb{K}^{(1)}$. An implication AxCI [5] is then a special case of a AxCAR since its confidence is 1. From Figures 2 and 3 one extract the AxCAR: R-c $\rightarrow$ P-aP-bP-dN-aN-bN-dR-aK-aK-bS-a $(0.2, 1)$.

In the following we define two types of triadic association rules that can be extracted from a triadic context $\mathbb{K} := (K_1, K_2, K_3, Y)$ by borrowing and combining ideas from [3,5]. The two types are: *conditional attribute* association rules and *attributional condition* association rules that we will call BCAARs and BACARs. respectively. We attach to such rules three quality measures: support, confidence and a new measure called *coverage*. The coverage of a given triadic association rule is computed as the ratio $|C|/|K_3|$ for a BCAAR and $|C|/|K_2|$ for an BACAR, and indicates the ratio of conditions (resp. attributes) for which an association between attributes (resp. conditions) holds.

3.3 Conditional Attribute Association Rules

A Biedermann conditional attribute association rule BCAAR has the form: $(A \rightarrow D)_C$ (s, c, cov). Its meaning is as follows: whenever A occurs under all conditions in C, then D also occurs under the same conditions with a support s, a confidence c and a coverage cov. By extending the definition of conditional attribute implication [5], the *conditional attribute* association rule CAAR $A \xrightarrow{C} D$ (s, c, cov) holds if $(A \rightarrow D)_{Ci}$ (s_i, c_i, cov) holds for all $Ci \subseteq C$ where s (resp. c) is the minimal support (resp. confidence) among the s_i (resp. c_i).

Figure 4 helps visualize and easily interpret a set of conditional attribute association rules with a confidence equal to 1. For example, the node (see the left part of the figure) whose extent contains N-P and whose successors are labeled with **a**ccessories and **d**igital cameras represents the implication $N \xrightarrow{ad} P$ $(0.4, 1, 0.5)$. It means that whenever **N**elson supplies any one of these two items (or both), then **P**eter also supplies such product(s). The implication $\emptyset \rightarrow P$ attached to the node representing the meet of nodes labeled with accessories and digital cameras means that supplier **P**eter supplies *accessories* and *digital cameras* to *every customer*.

The following proposition defines the way a *conditional attribute implication* and an *attributional condition implication* (with Biedermann's meaning) are computed.

Proposition 3. *Given a t-generator* (U_2, U_3) *of the pair* (A_2, A_3) *in the triadic concept* (A_1, A_2, A_3) *where* $A_1 \subseteq K_1$. *Then, the Biedermann conditional attribute implication BCAI :* $(U_2 \rightarrow A_2 \setminus U_2)_{U_3}$ *holds with a support equal to* $|A_1|/|K_1|$ *provided* $A_2 \setminus U_2 \neq \emptyset$. *In a dual way, the Biedermann attributional condition implication BACI:* $(U_3 \rightarrow A_3 \setminus U_3)_{U_2}$ *occurs with a support equal to* $|A_1|/|K_1|$ *provided* $A_3 \setminus U_3 \neq \emptyset$ *holds.*

For example the BCAI: $(NK \rightarrow P)_b$ and the BACI: $(b \rightarrow d)_{PN}$ are extracted from node #13 of Figure 3.

A Biedermann conditional attribute association rule BCAAR with confidence less than 1 can be defined as follows.

Proposition 4. *Given a t-generator* (U_2, U_3) *of the pair* (A_2, A_3) *in the triadic concept* (A_1, A_2, A_3) *such that* A_2 *is a maximal set in* $((U_2, U_3)^{(1)})^{(1)}$ *that contains* U_2*. Let the precedence order* $(B_1, B_2, B_3) \prec_1 (A_1, A_2, A_3)$ *holds. Then, the conditional attribute association rule BCAAR:* $(U_2 \rightarrow B_2 \setminus A_2)_{U_3}$ *(s, c, cov) holds with a support* $s = |B_1|/|K_1|$*, a confidence* $c = |B_1|/|A_1|$ *and a coverage* $cov = |U_3|/|K_3|$ *provided the following conditions hold: (i)* $B_2 \setminus A_2 \neq \emptyset$*, and (ii)* $U_2 \subset B_2$ *and* $U_3 \subseteq B_3$*.*

By imposing the maximality of A_2 we discard the attributes in B_2 that are deduced from U_2 with a confidence equal to 1 (see an example below).

3.4 Attributional Condition Association Rules

A Biedermann attributional condition association rule BACAR has the form: $(A \rightarrow D)_C$ (s, c, cov). Its meaning is as follows: whenever the conditions in A occur for all attributes in C, then the conditions in D also occur for the same attributes with a support s, a confidence c and a coverage cov.

Proposition 5. *Given a t-generator* (U_2, U_3) *of the pair* (A_2, A_3) *in the triadic concept* (A_1, A_2, A_3) *such that* A_3 *is a maximal set in* $((U_2, U_3)^{(1)})^{(1)}$ *that contains* U_3*. Let the precedence order* $(B_1, B_2, B_3) \prec_1 (A_1, A_2, A_3)$ *holds. Then, the Biedermann attributional condition association rule BACAR:* $(U_3 \rightarrow B_3 \setminus A_3)_{U_2}$ *(s, c, cov) holds with a support* $s = |B_1|/|K_1|$*, a confidence* $c = |B_1|/|A_1|$ *and a coverage* $cov = |U_2|/|K_2|$ *provided the following conditions are met: (i)* $B_3 \setminus A_3 \neq \emptyset$*, and (ii)* $U_2 \subseteq B_2$ *and* $U_3 \subset B_3$*.*

The rule $(U_3 \rightarrow B_3 \setminus A_3)_{U_2}$ (s, c, cov) becomes an attributional condition association rule ACAR $U_3 \xrightarrow{U_2} B_3 \setminus A_3$ (s, c, cov) if it holds for any subset in U_2. For example (see Figure 3), (NK, b) is a $t-$generator for $(A_2, A_3) := (PNK, b)$ while (PN, b) is a $t-$generator for both (PNK, b) and (PN, bd). The maximal value of A_2 w.r.t. both t-generators is PNK while the maximal value for A_3 w.r.t. the t-generator (PN, b) is bd. Since $(1, PN, abd) \prec_1 (14, PN, bd)$ (see the link between nodes #13 and #14) and $(4, PNRK, b) \prec_1 (14, PNK, b)$ (see the link between nodes #13 and #19) and the two conditions of the previous proposition are verified, we can generate e.g. the BCAAR (and also CAAR): $PN \xrightarrow{b} R$ (0.2, 0.50, 0.25) and the ACAR: $b \xrightarrow{PN} a$ (0.2, 0.50, 0.50) because the BACAR $(b \rightarrow a)_{C_1}$ holds for all $C_1 \subseteq \{P, N\}$.

The benefits of CAARs and ACARs over AxCARs is that they express in a very compact and more meaningful way a plethora of association rules by factorizing attributes or conditions, respectively. In the first case (i.e., CAARs), the focus is on attribute associations with reference to conditions under which they hold while in the second case (i.e., ACARs), the focus is on condition links with reference to attributes for which they hold. Moreover, the size of these sets is reduced compared to BCAARs and BACARs since they are stronger (and more restrictive) than the latter kind. However, their computation is more time consuming than the Biedermann's version due to the checking of $(A \rightarrow D)_c$ for all $c \in C$, mainly when C is large.

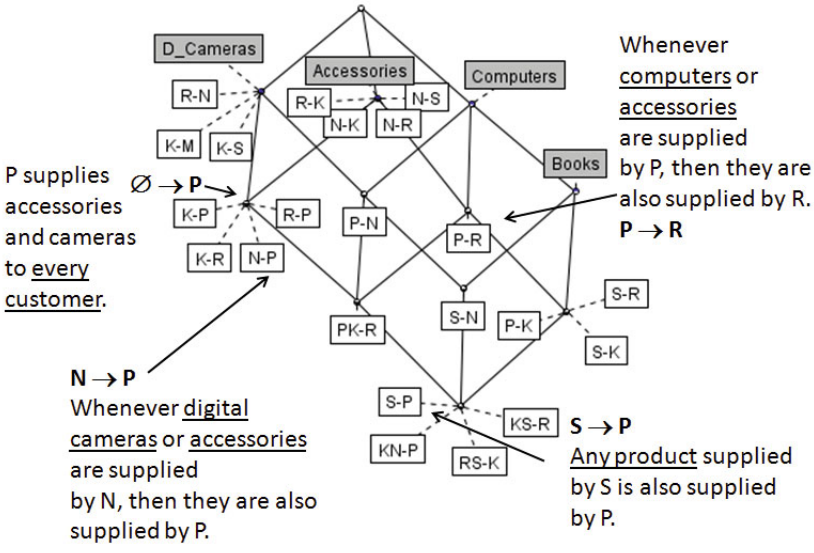


Fig. 4. The lattice of conditional attribute implications (CAIs) where e.g., N-P stands for $N \rightarrow P$

4 Triadic Pattern Computation

In the following we use the notion of generator to extract the generic base (containing only rules with confidence =1) and the informative base for approximate rule computation. As indicated earlier, our approach operates in three steps: (i) convert the triadic context into a dyadic one (e.g., $\mathbb{K}^{(1)} := (K_1, K_2 \times K_3, Y^{(1)})$), (ii) compute dyadic concepts, their corresponding (minimal) generators as well as *attribute*×*condition* association rules (AxCARs), and (iii) compute tri-sets, t-generators, and triadic association rules association rules (including implications). We will mainly focus on the Biedermann's meaning of implications and rules.

The main procedure is called TRIADIC (see Algorithm 1) and relies on other procedures to conduct all the needed computations. The concepts in the lattice L generated from $\mathbb{K}^{(1)}$ are first sorted according to a decreasing order of their extent size. This could be helpful for checking if (A_2, A_3) has already been retrieved with a greater extent (see line 18 of Algorithm 2). For each dyadic concept in the lattice, Procedure AXCAR (not shown in this paper) is called to compute AxCARs using the notions presented in Subsection 2.2. Then Algorithm 2 (line 8) is used to calculate the set C of triadic concepts together with their associated t-generators. If C is not empty (lines 9 to 11), then we add to TS (the set of $\sim_1$ classes containing tri-sets) a new class identified with its extent, its set C of pairs (A_2, A_3) and its t-generators in G. Once all triadic concepts (together with their associated t-generators) are produced, the link between classes of $\sim_1$ triadic concepts is established (see line 11) using for example the algorithm iPred [2]. Finally, the computation of BCAIs, BACIs, BCAARs, and BACARs is done in a polynomial time in the size of triadic concepts using Procedure TRAR (line 12) described by Algorithm 3.

Algorithm 1. Main procedure. Computation of triadic concepts and triadic association rules, including implications

Procedure TRIADIC(L)
In: L: a lattice of dyadic concepts described by their extent, intent, generators, successors and predecessors. K_1, K_2 and K_3 are global variables representing the object, attribute, and modus sets respectively.
Out: T: a set of $\sim_1$ classes containing triadic concepts and t-generators together with a precedence order between classes; Σ_1: a set of AxCARs; Σ_2: a set of BCAIs, BACIs, BCAARs and BACARs
$TS \leftarrow \emptyset$; $\Sigma_1 \leftarrow \emptyset$;
Sort(L)
for c in L **do**
  $\Sigma_1 \leftarrow \Sigma_1 \cup$ AxCAR(c) {Compute AxCARs from c}
  $(C, G) \leftarrow$ TRISET(c) {Compute the set C of triadic concepts and the set G of t-generators from c and its dyadic generators}
  if $C \neq \emptyset$ **then**
    $TS \leftarrow TS \cup \{(\text{Ext}(c), C, G)\}$ {Store $\sim_1$ tri-sets}
$T \leftarrow TS \cup$ LINK(TS)
$\Sigma_2 \leftarrow$ TRAR(T)
return T, Σ_1, Σ_2

Algorithm 2. Computation of triadic concepts and t-generators

Procedure TRISET(c)
In: c: a dyadic concept
Out: C and G: a set of triadic concepts and a set of associated t-generators
$C \leftarrow \emptyset$; $G \leftarrow \emptyset$
$U \leftarrow$ *Gen*(c) {*Gen*(c) is the set of generators associated with the intent of c}
$A \leftarrow$ *DistinctA*(c) {Collect distinct attribute values $a_j \in c$}
$M \leftarrow$ *DistinctM*(c) {Collect distinct modus values $a_k \in c$}
for $a_j \in A$ **do**
  for $a_k \in M$ **do**
    $K_{a_j,a_k} \leftarrow 0$ {Initialize the sub-context K_{a_j,a_k} to 0 where rows and columns are attributes and modus found in c.}
for e in Int(c) **do**
  $a_j \leftarrow$ *Attrib*(e) {Extract the attribute value a_j in $e = a_j \times a_k$ }
  $a_k \leftarrow$ *Mod*(e) {Extract the modus value a_k in e }
  $K_{a_j,a_k} \leftarrow 1$ {Construct the sub-context K_{a_j,a_k} from the intent of the concept c.}
$AM \leftarrow$ *AttMod*(K_{a_j,a_k}) {Compute the concepts (A_j, A_k) from K_{a_j,a_k}}
for $e \in AM$ **do**
  $e_1 \leftarrow$ *Derive*(e) {e_1 is the (1)-derivation of e}
  if $e_1 = \text{Ext}(c)$ **then**
    $C \leftarrow C \cup \{(e_1, \text{Ext}(e), \text{Int}(e))\}$
if $C \neq \emptyset$ **then**
  for g in U **do**
    $A \leftarrow$ *DistinctA*(g) {Collect distinct attribute values $a_j \in g$}
    $M \leftarrow$ *DistinctM*(g) {Collect distinct modus values $a_k \in g$}
    if $Size(A) \times \text{Size}(M) = \text{Size}(g)$ **then**
      $G \leftarrow G \cup \{(A, M)\}$
return C, G

Algorithm 2 exploits Propositions 1 and 2 to compute triadic concepts and t-generators when a dyadic concept c is given together with its associated (dyadic) generators. Lines 5 to 15 collect distinct attribute and modus values found in the intent of the current dyadic c in order to construct the sub-context K_{a_j,a_k} and generate the corresponding concepts. Line 18 checks whether each computed triple is a triadic concept. The computation of t-generators is done through lines 20 to 25. For each class of $\sim_1$ triadic concepts and their associated t-generators, Algorithm 3 uses Proposition 3 to first compute BCAIs and BACIs (lines 10 to 18) and then uses Propositions 4 and 5 to compute BCAARs and BACARs (lines 19 to 24) using Algorithm 4. In order to discard some redundant implications BCAIs (resp. BACIs), lines 20 and 23 select the closed tri-sets that have a maximal intent (resp. modus) containing the intent (resp. modus) of t-g.

To illustrate the execution of the different procedures, let us take the example given in Figure 3. From the dyadic concept c in node #13, Procedure AxCAR computes AxCARs like N-bK-b $\rightarrow$ P-aP-bP-dN-dR-aK-a $(0.4, 1)$ and N-bK-b $\rightarrow$ R-bS-d $(0.2, 0.5)$. Algorithm 2 computes the following elements: the set of dyadic generators $U =$ {P-bN-b, N-bK-b} as well as $A = \{P, N, R, K\}$ and $M = \{a, b, d\}$ corresponding to the set of attributes and conditions found in c, respectively. The set AM contains five couples (A_j, A_k), but only two tri-sets can be stored in $C = \{(14, PNK, b), (14, PN, bd)\}$. The set of t-generators to be extracted from U is $G = \{(NK, b), (PN, b)\}$. From Algorithms 3 and 4 a set of association rules can be computed such as $(NK \rightarrow R)_b$ $(0.2, 0.5, 0.25)$, $(NK \rightarrow P)_b$ $(0.4, 1, 0.25)$ and $(b \rightarrow a)_{NK}$ $(0.2, 0.5, 0.5)$.

5 Experimental Study

The objective of this section is to empirically estimate the execution time of the main procedure that computes both triadic concepts, t-generators and triadic association rules. To the best of our knowledge, there are neither implementations nor algorithms that handle the types of association rules that we are considering in this paper. Therefore, no empirical comparison between our procedures and other studies can be conducted. The empirical tests were done on a Windows XP-based system with 3 GB memory and 1.9 GHz processor on a pattern management environment using SQL Server. We generated five synthetic data sets of dyadic concepts obtained from triadic contexts of varying sizes. The size of the five datasets are 2197, 4350, 6289, 8461, and 10234 dyadic concepts. The execution time shown in Figure 5 includes the time needed to first generate dyadic concepts and then to compute triadic concepts, generators and rules from dyadic concepts. Other tests show that the overall cost is dominated by the computation of triadic concepts. We have also compared the size of AxCARs against the size of BCAARs and BACARs and found that the former ones are more numerous than the latter ones. As an example, from a set of 2197 concepts, we got 19107 AxCARs from which only 309 rules (232 BCAARs and 77 BCAIs) were produced. This result illustrates the potential of triadic rules to be more meaningful and compact than dyadic ones.

Further experimentation and implementation will be conducted in order to compare our present approach against other variants that we are developing (e.g., computing

Algorithm 3. Computation of triadic association rules, including implications

Procedure TRAR(T)
In: T= a set of classes. Each class contains $\sim_1$ triadic concepts together with associated t-generators and predecessors.
Out: $TRAR$: A set of n-uples (L, R, C, t, s, c, cov) representing association rules with left hand-side L, right hand-side R, condition C, type t (= 1 to 4 for BCAI, BACI, BCAAR and BACAR resp.), and quality measures.
$TRAR \leftarrow \emptyset$
for CL in T **do**
  for t-g in Gen(CL) **do**
    {Gen(CL) is the set of t-generators in the class CL of T}
    $A \leftarrow$ Int(t-g); $D \leftarrow$ Modus(t-g)
    for t-c in CL **do**
      {Compute BCAIs and BACIs}
      $MaxTriAtt \leftarrow$ MAX(t-c, Int(t-g)); $MaxTriMod \leftarrow$ MAX(t-c, *Modus*(t-g))
      $B \leftarrow$ Int(t-c); $F \leftarrow$ Modus(t-c); $E \leftarrow$ Ext(t-c);
      $s \leftarrow Size(E)/Size(K_1)$
      if t-c in $MaxTriAtt$ and $A \subset B$ and $D \subseteq F$ **then**
        $cov \leftarrow Size(D)/Size(K_3)$
        $TRAR \leftarrow TRAR \cup \{(A, B \setminus A, D, 1, s, 1, cov)\}$
      if t-c in $MaxTriMod$ and $D \subset F$ and $A \subseteq B$ **then**
        $cov \leftarrow Size(A)/Size(K_2)$
        $TRAR \leftarrow TRAR \cup \{(D, F \setminus D, A, 2, s, 1, cov)\}$
      {Compute BCAARs}
      if $Pred(CL) \neq \emptyset$ and t-c is maximal w.r.t. the intent A of t-g **then**
        $TRAR \leftarrow TRAR \cup$ AR($CL, A, D, E, 3$)
      {Compute BACARs}
      if $Pred(CL) \neq \emptyset$ and t-c is maximal w.r.t. the modus D of t-g **then**
        $TRAR \leftarrow TRAR \cup$ AR($CL, A, D, E, 4$)
return $TRAR$

Algorithm 4. Computation of BCAARs and BACARs

Procedure AR(CL, A, D, E, i)
In: CL: A class of $\sim_1$ triadic concepts together with associated t-generators and predecessors; A and D are the intent and the modus of the current gt-generator respectively while E is the extent of the current triadic concept and i takes de value 3 or 4
Out: $Temp$: a set of association rules whose confidence is less than 1.
$Temp \leftarrow \emptyset$
for p in $Pred(CL)$ **do**
  $B \leftarrow$ Int(p); $F \leftarrow$ Modus(p)
  $s \leftarrow$ Ext$(p)/Size(K_1)$
  $c \leftarrow$ Ext$(p)/Size(E)$
  if $i = 3$ and $A \subset B$ and $D \subseteq F$ **then**
    $cov \leftarrow Size(D)/Size(K_3)$
    $Temp \leftarrow Temp \cup \{(A, B \setminus A, D, 3, s, c, cov)\}$
  if $i = 4$ and $D \subset F$ and $A \subseteq B$ **then**
    $cov \leftarrow Size(A)/Size(K_2)$
    $Temp \leftarrow Temp \cup \{(D, F \setminus D, A, 4, s, c, cov)\}$
return $Temp$

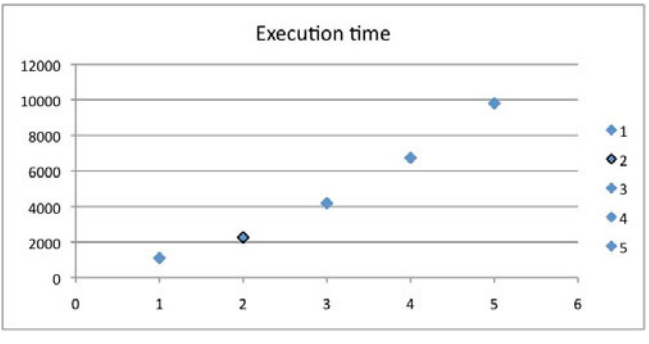


Fig. 5. Execution time according to the number of dyadic concepts

triadic concepts and generators directly from the initial triadic context) and analyze the effects of context configuration (e.g., density, number of attributes) on performance.

6 Conclusion

In this work, we defined procedures for association rule mining using triadic concept analysis as a theoretical framework. A couple of issues need to be explored: (i) alternate solutions towards a more efficient computation of triadic concepts, generators and association rules, (ii) incremental computation of triadic concepts when one of the three sets is augmented, and (iii) computation of n-ary association rules by exploring existing studies on generating n-ary patterns (concepts and rules) from polyadic contexts [4,18].

Acknowledgment

The first author acknowledges the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC). The two authors would like to warmly thank anonymous referees for raising important questions and making relevant suggestions that helped improve the quality of the paper.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB, pp. 487–499 (1994)
2. Baixeries, J., Szathmary, L., Valtchev, P., Godin, R.: Yet a faster algorithm for building the hasse diagram of a concept lattice. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS, vol. 5548, pp. 162–177. Springer, Heidelberg (2009)
3. Biedermann, K.: How triadic diagrams represent conceptual structures. In: ICCS 1997, pp. 304–317 (1997)
4. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.-F.: Closed patterns meet -ary relations. TKDD 3(1) (2009)
5. Ganter, B., Obiedkov, S.A.: Implications in triadic formal contexts. In: ICCS, pp. 186–195 (2004)
6. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag New York, Inc., Heidelberg (1999) (Translator-C. Franzke)
7. Guigues, J.-L., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines 95(1), 5–18 (1986)
8. Hamrouni, T., Valtchev, P., Yahia, S.B., Nguifo, E.M.: About the lossless reduction of the minimal generator family of a context. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 130–150. Springer, Heidelberg (2007)
9. Jäschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Trias - an algorithm for mining iceberg tri-lattices. In: ICDM, pp. 907–911 (2006)
10. Ji, L., Tan, K.-L., Tung, A.K.H.: Mining frequent closed cubes in 3d datasets. In: VLDB, pp. 811–822 (2006)
11. Kryszkiewicz, M., Gajek, M.: Concise representation of frequent patterns based on generalized disjunction-free generators. In: Chen, M.-S., Yu, P.S., Liu, B. (eds.) PAKDD 2002. LNCS (LNAI), vol. 2336, pp. 159–171. Springer, Heidelberg (2002)
12. Lehmann, F., Wille, R.: A triadic approach to formal concept analysis. In: ICCS, pp. 32–43 (1995)
13. Luxenburger, M.: Implications partielles dans un contexte. Mathématiques, Informatique et Sciences Humaines 29(113), 35–55 (1991)
14. Nguyen, K.N.T., Cerf, L., Plantevit, M., Boulicaut, J.-F.: Discovering inter-dimensional rules in dynamic graphs. In: Proc. Workshop on Dynamic Networks and Knowledge Discovery DYNAK 2010 co-located with ECML/PKDD 2010, Barcelona, pp. 5–16 (2010)
15. Nourine, L., Raynaud, O.: A fast incremental algorithm for building lattices. J. Exp. Theor. Artif. Intell. 14(2-3), 217–227 (2002)
16. Pasquier, N., Bastide, Y., Taouil, T., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Information Systems 24(1), 25–46 (1999)
17. Szathmary, L., Valtchev, P., Napoli, A., Godin, R.: Constructing iceberg lattices from frequent closures using generators. In: Boulicaut, J.-F., Berthold, M.R., Horváth, T. (eds.) DS 2008. LNCS (LNAI), vol. 5255, pp. 136–147. Springer, Heidelberg (2008)
18. Voutsadakis, G.: Polyadic concept analysis. Order 19(3), 295–304 (2002)
19. Wille, R.: The basic theorem of triadic concept analysis. Order 12(2), 149–158 (1995)

The Agree Concept Lattice for Multidimensional Database Analysis

Sébastien Nedjar, Fabien Pesci, Lotfi Lakhal, and Rosine Cicchetti

Laboratoire d'Informatique Fondamentale de Marseille (LIF), CNRS UMR 6166
Aix-Marseille Université, IUT d'Aix-en-Provence,
Avenue Gaston Berger, 13625 Aix-en-Provence Cedex

Abstract. In this paper we propose the characterization of two new structures, the Agree Concept Lattice and the Quotient Agree Lattice of a database relation. Both of them are of great interest for multidimensional database analysis. They provide a formal framework which makes it possible to improve computation time, reduce representation and easily navigate through the Hasse diagram. These structures are generic, apply to various database analysis problems and combine formal concept analysis and database theory. They make use of the concepts of agree set and database partition. Agree set and partition are associated to define the Agree Concept of a database relation. The set of all the Agree Concepts is organized within the Agree Concept Lattice. The Quotient Agree Lattice is along the lines of both the TITANIC framework and the quotient cube.

We also briefly present three application fields of the proposed structures. The first two ones are classical since they concern on the one hand the discovery of functional and approximate dependencies for database design and tuning and on the other hand the data cube computation and representation. The latter field has been recently investigated. The underlying issue is to retrieve the most relevant objects according to the user expectations: the SKYLINE. The multidimensional generalization of the SKYLINE has been proposed through the SKYCUBE. The proposed structures smartly solve the problem of partial materialization of SKYCUBE with reconstruction guarantee.

Keywords: Agree set, Database partition, Concept lattice, OLAP mining, Multidimensional database analysis.

1 Introduction

In the fields of databases and data warehouses, several very different issues require manipulating very voluminous data sets, performing costly computations and storing overwhelming volumes of results. Among such issues, let us quote the extraction of functional or approximate dependencies (Lopes et al., 2002), the computation and representation of data cubes (Casali et al., 2009a; Nedjar et al., 2009; Casali et al., 2003; Lakshmanan et al., 2002) and the multidimensional database analysis through the SKYCUBE concept (Pei et al., 2006,

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 219–234, 2011.

2005; Yuan et al., 2005). In this paper, we propose two well founded and generic structures which can be used to solve the given issues. They associate formal concept analysis (Ganter and Wille, 1999) and database theory (Abiteboul et al., 1995). The former has been successfully used when addressing various issues in database, data mining and data warehousing. For instance, different approaches have been defined to extract frequent closed item-sets (Pasquier et al., 1999; Stumme et al., 2002) and association rules, represent such rules in a compact way through basis or covers (Zaki, 2000; Bastide et al., 2000; Pasquier et al., 2005), discover functional and conditional dependencies (Lopes et al., 2000; Novelli and Cicchetti, 2001; Diallo et al., 2011; Medina and Nourine, 2010), reduce the size of data cubes by representing them through constrained closed or quotient cubes (Nedjar et al., 2010b, 2009, 2010c,a). The former two quoted approaches mine new knowledge from transaction databases or formal contexts. For the latter ones, knowledge extraction is performed from a database relation which represents a many-valued context. All these approaches take advantage of the formal concept analysis expressiveness in order to soundly characterize the tackled problems, devise efficient algorithms and propose reduced representations as well as visual navigation tools through the solution space. Our proposals fit in a similar spirit. We characterize the Agree Concept Lattice and the Quotient Agree Lattice. From database theory, we make use of the concepts of partition and agree set. The partition of a relation according to a set of attributes X is a set of parts (or equivalence classes) in which all the stored objects (or tuples) share the same value for X. An agree set is a set of attributes for which certain tuples agree, *i.e.* share the same value (Beeri et al., 1984; Lopes et al., 2002). The agree sets and partitions are combined through a particular Galois connection in order to constitute Agree Concepts. The set of all the Agree Concepts is provided with a twofold order relationship: inclusion between attribute sets and refinement between database partitions. The result is the Agree Concept Lattice. When compared to this lattice, the Quotient Agree Lattice associates to each Agree Concept intent all its minimal generators in order to represent equivalence classes. These classes are built up according to an equivalence relationship for which we define three particular and equivalent instances. We also briefly describe two classical application fields: the approximate and functional dependency discovery and the data cube computation and representation. Moreover we investigate the most promising and innovating application field: the multidimensional and multicriterion database analysis based on Skylines (Börzsönyi et al., 2001). The Skyline operator considers the set of the chosen user criteria as preferences and extracts the optimal objects for this set. It is based on the notion of dominance (Pareto's dominance relationship (Kung et al., 1975)). The Skycube (Yuan et al., 2005; Pei et al., 2005) groups all the Skylines according to the possible subsets of criteria (or attributes). With the two proposed structures, we are able to define a partial materialization approach which makes it possible to reconstruct the whole Skycube.

The paper is organized as follows. In Section 2, we remind our background: the concepts of agree set and partition. The two following sections are devoted

to the two new lattice structures. Finally we place emphasis on their application in several database analysis problems.

2 Background

In this section, we present the fundamental concepts for our approach: the partition lattice (Birkhoff, 1970) and agree set (Beeri et al., 1984) originated from database theory.

2.1 The Partition Lattice

The concepts reminded in this section are classical in mathematics and have been used for solving database problems (Spyratos, 1987).

Definition 1 (Partition of a set). *Let E be a set, a partition $\pi(E)$[1] of the set E is a family of parts of this set such that each element of E exactly belongs to a single part (also called class). In other words, $\pi(E)$ is a family of disjointed sets ($\forall X, Y \in \pi(E)$ we have $X \cap Y = \emptyset$) and their union is equal to E ($\bigcup_{X \in \pi(E)} = E$).*

Definition 2 (Order relationship between partitions). *Let $\pi(E)$, $\pi'(E)$ be two partitions of a single set E, $\pi(E)$ is a refinement of $\pi'(E)$ if and only if any class of $\pi(E)$ is obtained by dividing classes of $\pi'(E)$[2]. The refinement relationship between two partitions is a partial order relation noted $\sqsubseteq$. It is defined as follows:*

$$\begin{aligned} \pi(E) \sqsubseteq \pi'(E) &\Leftrightarrow \pi(E) \text{ is a refinement of } \pi'(E)^{3} \\ &\Leftrightarrow \forall X \in \pi(E),\ \exists X' \in \pi'(E),\ X \subseteq X' \end{aligned}$$

From this definition, we have $\pi(E) \sqsubseteq \pi'(E) \Rightarrow |\pi(E)| \geq |\pi'(E)|$.

Definition 3 (Partition product). *Let $\pi(E)$ and $\pi'(E)$ be two partitions of a single set E. The product of the partitions $\pi(E)$ and $\pi'(E)$, noted $\pi(E) \bullet \pi'(E)$, is obtained as follows:*

$$\pi(E) \bullet \pi'(E) = \{Z = X \cap Y \mid Z \neq \emptyset,\ X \in \pi(E) \text{ and } Y \in \pi'(E)\}$$

Before defining the operator sum between two partitions, we introduce a tool function $\mathcal{U}$:

$$\mathcal{U}(e, F) = \bigcup_{\substack{X \in F \\ e \in X}} X$$

with e an element of a set E, F a family of parts of E. $\mathcal{U}(e, F)$ corresponds to the union of sets of F containing the element e.

[1] When there is no ambiguity for the set E we note the partition π.

[2] In an equivalent way, $\pi(E)$ is a refinement of $\pi'(E)$ if and only if any class of $\pi'(E)$ results from the union of classes of $\pi(E)$.

[3] $\pi'(E)$ is said rougher than $\pi(E)$.

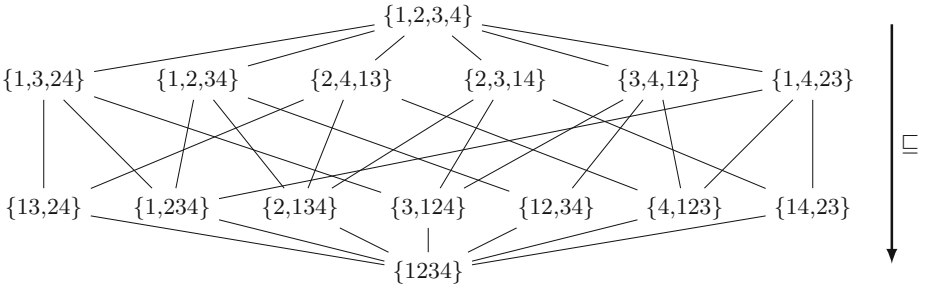


Fig. 1. Hasse Diagram of the partition lattice of $E = \{1, 2, 3, 4\}$

Definition 4 (Sum of partitions). *Let $\pi(E)$ and $\pi'(E)$ be two partitions of a single set E. The sum of the partitions $\pi(E)$ and $\pi'(E)$, noted $\pi(E) + \pi'(E)$, is obtained by the transitive closure of the operation which associates to an element of E the set of elements of its classes in $\pi(E)$ and $\pi'(E)$ (Barbut, 1968). The sequence S is defined below in order to characterize this computation :*

$$\begin{cases} S_0 = \max\limits_{\subseteq}(\pi(E) \cup \pi'(E)) \\ S_n = \max\limits_{\subseteq}(\{\mathcal{U}(e, S_{n-1}) \mid e \in E\}) \end{cases}$$

Thus the operator sum can be defined as follows:

$$\pi(E) + \pi'(E) = S_k \text{ with } k \text{ such that } S_k = S_{k-1}$$

Theorem 1 (Partition lattice). *Let $\Pi(E)$ be the set of the possible partitions of a set E. The ordered set $\langle \Pi(E), \sqsubseteq \rangle$ forms a complete lattice called partition lattice of E. $\forall P \subseteq \Pi(E)$, its* infimum *or lower bound ($\bigwedge$) and its* supremum *or upper bound ($\bigvee$) are given below:*

$$\bigwedge P = \bigbullet_{\pi \in P} \pi, \qquad \bigvee P = \bigplus_{\pi \in P} \pi$$

Example 1. The Hasse diagram of the partition lattice of $E = \{1, 2, 3, 4\}$ is given in Figure 1. In order to keep the uniformity with the Agree Concept Lattice, the lattice is represented in a reverse way in comparison with the classical representation. The roughest partitions are at the bottom and the thinnest ones at the top.

2.2 Agree Sets

The concept of agree set (as well as the associated closure system), introduced in Beeri et al. (1984) to characterize the Armstrong relation, has been successfully used to discover exact and approximate functional dependencies (Lopes et al., 2002). Two tuples agree on an attribute set X if they share the same value on X.

Table 1. The relation HOUSING

RowId	Price	Distance	Consumption	Neighbors
1	220	15	275	5
2	100	15	85	1
3	150	7	180	1
4	340	7	85	3
5	100	7	180	1

Definition 5 (Agree set). *Let r be a database relation with the set of attributes $\mathcal{R}$ and t_i, t_j two tuples of r. $X \subseteq \mathcal{R}$ is an attribute set. t_i, t_j agree on X if and only if $t_i[X] = t_j[X]$. The agree set of t_i and t_j is defined as follows:*

$$\text{ACC}(t_i, t_j) = \{X \subset \mathcal{R} \mid t_i[X] = t_j[X]\}$$

This definition can be generalized to a set of tuples $T \subseteq r$ encompassing at least two elements:

$$\text{ACC}(T) = \{X \subset \mathcal{R} \mid t[X] = t'[X],\ \forall t, t' \in T\}$$

Definition 6 (Agree set of a database relation). *The agree set of a database relation r is defined as follows:*

$$\text{AGREE}(r) = \{\text{ACC}(t_i, t_j) \mid t_i, t_j \in r \text{ and } i \neq j\}$$

This set can be defined in an equivalent way:

$$\text{AGREE}(r) = \{\text{ACC}(T) \mid \forall T \subseteq r \text{ and } |T| \geq 2\}$$

Example 2. The relation depicted in Table 1 lists various housings. The attributes are: the sale *Price* in thousands of euros, the *Distance* from the work place, the energy *Consumption* in kilowatt-hours by year and square meter, the number of *Neighbors*. The attributes are denoted by their initial.

$\text{ACC}(t_2, t_5) = PN$ because these two tuples share the same value for the attributes *Price*, *Neighbor* and are provided with different values for *Distance*, *Consumption*. In the same way, $\text{ACC}(\{t_3, t_4, t_5\}) = D$ because these three tuples have the same value only for the attribute *Distance*. The agree attribute set of the relation HOUSING is the following:
$\text{AGREE}(\text{HOUSING}) = \{\emptyset, D, N, C, PN, DCN\}$.

Definition 7 (Equivalence class of a tuple). *Let r be a relation and $X \subseteq \mathcal{R}$ a set of attributes. The class of a tuple t according to X, noted $[t]_X$, is defined as the set of identifiers i (Rowid) of all the tuples $t_i \in r$ which agree with t according to X* (i.e. *the identifier set of the tuples t_i sharing with t the same values for X). Thus we have:*

$$[t]_X = \{i \in Tid(r)^4 \mid t_i[X] = t[X]\}$$

[4] $Tid(r) = \{Rowid(t) \mid t \in r\}$.

Example 3. With the relation HOUSING, $[t_2]_P = \{2, 5\}$ because t_2 and t_5 are provided with the same value for the attribute *Price*.

3 Agree Concept Lattice of a Database Relation

In this section, our objective is to define a formal framework combining the concepts of agree set and the one of concept lattice. We propose a new structure, the Agree Concept Lattice of a relation, which can be used to solve several multidimensional database analysis problems. Then we soundly characterize the Agree Concept Lattice.

3.1 Agree Concepts of a Database Relation

Our objective is to define a particular concept lattice which is based on the agree sets and the database partitions (Spyratos, 1987). In order to meet this goal, we characterize an instance of the Galois connection between on the one hand the lattice of the power set of the attribute set and on the other hand the lattice of the partitions of the tuple identifier set. This connection makes it possible to define dual closure operators, introduce the Agree Concept and characterize the Agree Concept Lattice.

Definition 8. *Let Rowid* : $r \rightarrow \mathbb{N}$ *be an application which associate to each tuple a single natural integer and* $Tid(r) = \{Rowid(t) \mid t \in r\}$. *Let* f, g *be two applications between the ordered sets* $\langle \Pi(Tid(r)), \sqsubseteq \rangle$ *and* $\langle \mathcal{P}(\mathcal{R})$[5]$, \subseteq \rangle$ *which are defined as follows:*

$$
\begin{array}{rcl}
f : \langle \Pi(Tid(r)), \sqsubseteq \rangle & \longrightarrow & \langle \mathcal{P}(\mathcal{R}), \subseteq \rangle \\
\pi & \longmapsto & \bigcap_{[t] \in \pi} \textsc{Acc}(\{t_i \mid i \in [t]\}) \\
g : \langle \mathcal{P}(\mathcal{R}), \subseteq \rangle & \longrightarrow & \langle \Pi(Tid(r)), \sqsubseteq \rangle \\
X & \longmapsto & \{[t]_X \mid t \in r\}
\end{array}
$$

For an attribute set X, the equivalence class set according to X forms a partition of $Tid(r)$. It is the function g which associates this partition of identifiers to X. This partition is noted π_X and defined as follows: $\pi_X = g(X)$. The set of all the possible partitions π_X is noted $\Pi_{\mathcal{P}(\mathcal{R})}$(Lopes et al., 2000; Laporte et al., 2002). The function f performs the opposite association.

Example 4. With the relation HOUSING, by considering the following attribute set: D, DC, DCN and PN. we have $g(D) = \{12, 345\}$[6], $g(DC) = \{1, 2, 35, 4\}$, $g(DCN) = \{1, 2, 35, 4\}$ and $g(PN) = \{1, 25, 3, 4\}$. With the partitions $\{1, 2, 35, 4\}$ and $\{1, 2, 345\}$, we have $f(\{1, 2, 35, 4\}) = DCN$ and $f(\{1, 2, 345\}) = D$.

Proposition 1. *The couple of applications* $gc = (f,\ g)$ *is a Galois connection between the attribute power set lattice of* $\mathcal{R}$ *and the lattice of partitions of* $Tid(r)$.

[5] $\mathcal{P}(\mathcal{R})$ is the powerset lattice of the attribute set of the database relation r.
[6] t_1 and t_2 are in the same equivalence class. t_3, t_4 and t_5 belong to the same class.

Proof. Due to the definition of the order relationship between partitions and the definition of f, it is trivial to show that $\pi \sqsubseteq \pi_X \Leftrightarrow X \subseteq f(\pi)$. According to Ganter and Wille (1999), $gc = (f,\ g)$ is a Galois connection. □

Definition 9 (Closure operators). *The couple $gc = (f,\ g)$ is a particular case of the Galois connection. The compositions $f \circ g$ and $g \circ f$ of the two applications are closure operators (Ganter and Wille, 1999). They are defined below:*

$$h : \quad \mathcal{P}(\mathcal{R}) \longrightarrow \mathcal{P}(\mathcal{R})$$
$$X \longmapsto f(g(X)) = \bigcap_{\substack{X' \in \text{AGREE}(r) \\ X \subseteq X'}} X'$$
$$h' : \Pi(Tid(r)) \longrightarrow \Pi(Tid(r))$$
$$\pi \longmapsto g(f(\pi)) = \bullet_{\substack{\pi' \in \Pi_{\mathcal{P}(\mathcal{R})} \\ \pi \sqsubseteq \pi'}} \pi'$$

h and h' satisfy the following properties:

1. $X \subseteq X' \Rightarrow h(X) \subseteq h(X')$ and $\pi \sqsubseteq \pi' \Rightarrow h'(\pi) \sqsubseteq h'(\pi')$ (monotony)
2. $X \subseteq h(X)$ et $\pi \sqsubseteq h'(\pi)$ (extensivity)
3. $h(X) = h(h(X))$ et $h'(\pi) = h'(h'(\pi))$ (idempotence)

Example 5. With the relation HOUSING, by considering the attribute sets DC and DCN, according to the previous example, we have:

- $h(DC) = f(g(DC)) = f(\{1, 2, 35, 4\}) = DCN$
- $h(DCN) = f(g(DCN)) = f(\{1, 2, 35, 4\}) = DCN$

With the partitions $\{1, 2, 35, 4\}$ and $\{1, 2, 345\}$, we have:

- $h'(\{1, 2, 35, 4\}) = g(f(\{1, 2, 35, 4\})) = g(DCN) = \{1, 2, 35, 4\}$
- $h'(\{1, 2, 345\}) = g(f(\{1, 2, 345\})) = g(D) = \{12, 345\}$

Definition 10 (Agree concepts). *An Agree concept of a database relation r is a couple (X, π) associating a set of attributes to a partition of identifiers: $X \in \mathcal{P}(\mathcal{R})$ and $\pi \in \Pi(Tid(r))$. The elements of this couple must be related by the following conditions : $X = f(\pi)$, $\pi = g(X) = \pi_X$.*
Let $c_a = (X_{c_a}, \pi_{c_a})$ an Agree Concept of r, we call π_{c_a} the extent *of c_a (noted $ext(c_a)$) and C_{c_a} its* intent *(noted $int(c_a)$). The set of all the Agree Concepts of a relation r is noted* AGREECONCEPTS(r).

Theorem 2 (Agree Concept Lattice). *Let* AGREECONCEPTS(r) *be the set of Agree Concepts of a relation r. The ordered set $\langle$*AGREECONCEPTS$(r), \leq$[7]$\rangle$ *forms a complete lattice called the Agree Concept lattice. $\forall P \subseteq$* AGREECONCEPTS(r),

[7] Let (X_1, π_1), $(X_2, \pi_2) \in$ AGREECONCEPTS(r), $(X_1, \pi_1) \leq (X_2, \pi_2) \Leftrightarrow X_1 \subseteq X_2$ (or in an equivalent way $\pi_2 \sqsubseteq \pi_1$).

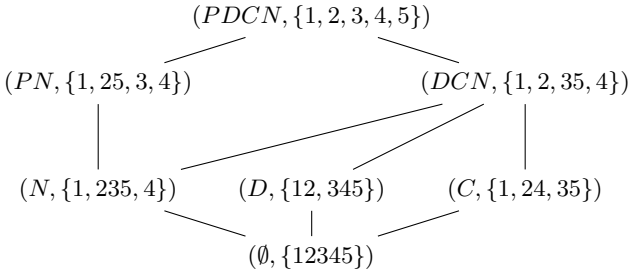


Fig. 2. Hasse diagram of the Agree Concept Lattice of the relation HOUSING

the infimum *or lower bound* ($\bigwedge$) *and* supremum *or upper bound* ($\bigvee$) *are given below:*

$$\bigwedge P = (\bigcap_{c_a \in P} int(c_a),\ h'(\underset{c_a \in P}{+} ext(c_a)))$$

$$\bigvee P = (h(\bigcup_{c_a \in P} int(c_a)),\ \underset{c_a \in P}{\bullet} ext(c_a))$$

Proof. Since the couple $gc = (f,\ g)$ is a Galois connection, the Agree Concept Lattice is a concept lattice according to the fundamental theorem of Wille (Ganter and Wille, 1999).

Example 6. Figure 2 gives the Hasse diagram of the Agree Concept lattice of the relation HOUSING. The couple $(DCN, \{1, 2, 35, 4\})$ is an Agree Concept because according to the examples 4 we have $g(DCN) = \{1, 2, 35, 4\}$ and $f(\{1, 2, 35, 4\}) = DCN$. In contrast, the couple $(DC, \{1, 2, 35, 4\})$ is not an Agree Concept because $f(\{1, 2, 35, 4\}) \neq DC$. Let $c_a = (DCN, \{1, 2, 35, 4\})$ and $c_b = (PN, \{1, 25, 3, 4\})$ be two Agree Concepts, thus we have:

$$\begin{aligned} c_a \wedge c_b &= (DCN \cap PN,\ h'(\{1, 2, 35, 4\} + \{1, 25, 3, 4\})) \\ &= (N,\ h'(\{1, 235, 4\})) = (N,\ \{1, 235, 4\}) \\ c_a \vee c_b &= (h(DCN \cup PN),\ \{1, 2, 35, 4\} \bullet \{1, 25, 3, 4\}) \\ &= (h(PDCN),\ \{1, 2, 3, 4, 5\}) = (PDCN,\ \{1, 2, 3, 4, 5\}) \end{aligned}$$

Proposition 2. *For any attribute set $X \subseteq \mathcal{R}$, the associated partition π_X is identical to the partition of its closure.*

$$\forall X \subseteq \mathcal{R}, \pi_X = \pi_{h(X)}$$

Proof. By definition $\forall X \subseteq \mathcal{R}$, $\pi_X = g(X)$ and $h(X) = f(g(X))$. Thus we have $\pi_{h(X)} = g(f(g(X))$. Since the couple $gc = (f,\ g)$ is a Galois connection, we have $g \circ f \circ g = g$ (Ganter and Wille, 1999). Then, $\pi_{h(X)} = g(f(g(X))) = g(X) = \pi_X$.

Example 7. With the relation HOUSING, by considering the set of attributes DC, according to the examples 4 and 5, we have: $\pi_{DC} = g(DC) = \{1, 2, 35, 4\}$ and $\pi_{h(DC)} = g(h(DC)) = g(DCN) = \{1, 2, 35, 4\}$.

The previous proposition means that the closure of a set of attributes X can be seen as the greatest super-set of X provided with the very same partition.

4 Quotient Agree Lattice of a Database Relation

In this section, we define the Quotient Agree Lattice which is inspired by the structure of the quotient cube (Lakshmanan et al., 2002) which itself is along the lines of TITANIC (Stumme et al., 2002).

The idea behind the structure in question is to discard redundancies by gathering together elements sharing an equivalent information. This results in a set of equivalence classes partitioning the original set. Such a partitioning can be performed in various ways. However, in order to preserve the navigation capabilities through the structure, it is important to deal with convex classes.

Definition 11 (Convex equivalence class). *Let $\langle E, \leq \rangle$ be an ordered set and $\mathcal{C} \subseteq E$ be an equivalence class. We say $\mathcal{C}$ is convex if and only if:*

$$\forall e \in E \text{ if } \exists e', e'' \in \mathcal{C} \text{ such that } e' \leq e \leq e'' \text{ then } e \in \mathcal{C}$$

A partition $\mathcal{P}$ of E which only encompasses convex equivalence classes is called a convex partition.

The convexity property makes it possible to represent each equivalence class through its maximal and minimal tuples. Intermediary tuples are no longer useful and the underlying representation is reduced. To ensure that the partition is convex (Lakshmanan et al., 2002), the following equivalence relation is used.

Definition 12 (Quotient equivalence relation). *The equivalence relation $\equiv$ is said a* quotient equivalence relation *if and only if it satisfies the property of weak congruence:*

$$\forall e, e', f, f' \in E, \text{ if } e \equiv e', f \equiv f', e \leq f \text{ and } f' \leq e', \text{ then } e \equiv f$$

We denote $[e]_{\equiv}$ the equivalence class of $e \in E$ defined by : $[e]_{\equiv} = \{e' \in E \mid e \equiv e'\}$

The construction of a quotient lattice depends on the chosen quotient equivalence relation. As a consequence, two different quotient equivalence relations result in two different quotient lattices. In order to define the Quotient Agree Lattice of a database relation, we give the following equivalence relation.

Definition 13 (π-equivalence relation). *Let $X, Y \in \mathcal{P}(\mathcal{R})$ be two sets of attributes. X, Y are π-equivalent over r, $X \equiv_\pi Y$, if they have the same partition.*

$$\forall X, Y \in \mathcal{P}(\mathcal{R}), \; X \equiv_\pi Y \Leftrightarrow \pi_X = \pi_Y$$

Example 8. With the relation HOUSING, by considering the attribute set DC, according to the example 7, we have: $\pi_{DC} = \{1, 2, 35, 4\}$ and $\pi_{DCN} = \{1, 2, 35, 4\}$ hence $DC \equiv_\pi DCN$.

Proposition 3. *the π-equivalence relation is a quotient equivalence relation.*

Proof.

$$\begin{aligned}\forall X, X', Y, Y' \in \mathcal{P}(\mathcal{R}),\ X \equiv_\pi X',\ Y \equiv_\pi Y',\ X \subseteq Y \text{ and } Y' \subseteq X' \\ \Leftrightarrow \pi_X = \pi_{X'},\ \pi_Y = \pi_{Y'},\ \pi_Y \sqsubseteq \pi_X,\ \pi_{X'} \sqsubseteq \pi_{Y'} \\ \Rightarrow \pi_Y \sqsubseteq \pi_X,\ \pi_X \sqsubseteq \pi_Y \Rightarrow \pi_Y = \pi_Y \Rightarrow X \equiv_\pi Y\end{aligned}$$

□

Using the π-equivalence relation as a quotient equivalence relation, we are able to define the *Quotient Agree Lattice*. It is defined as the set of equivalence classes of $\mathcal{P}(\mathcal{R})$, each one being provided with its associated partition.

Definition 14 (Quotient Agree Lattice). *Let $\equiv_\pi$ be the π-equivalence relation. The Quotient Agree Lattice of r, denoted by $QuotientAgreeLattice(r)$, is defined as follows:*

$$QuotientAgreeLattice(r) = \{([X]_{\equiv_\pi}, \pi_X) \text{ such that } X \in \mathcal{P}(\mathcal{R})\}$$

*The Quotient Agree Lattice of r is a convex partition of $\mathcal{P}(\mathcal{R})$ (*cf. *proposition 3 and definition 12). Each equivalence class of the Quotient Agree Lattice which contains more than one element is represented by its maximal element (w.r.t inclusion) which is an Agree Concept and its minimal elements which are the minimal generators associated with the quoted Agree Concept. Equivalence classes such that $|[X]_{\equiv_\pi}| = 1$ are represented by their single element.*

For two equivalence classes $\mathcal{C}, \mathcal{C}' \in QuotientAgreeLattice(r)$, $\mathcal{C} \preceq_Q \mathcal{C}'$ when $\exists X \in \mathcal{C}$ and $\exists X' \in \mathcal{C}'$ such that $X \subseteq X'$.

Example 9. Figure 3 gives the Hasse diagram of the Quotient Agree Lattice of the relation HOUSING.

Definition 15 (count equivalence relation). *Let $X, Y \in \mathcal{P}(\mathcal{R})$ be two sets, the count equivalence relation over r, $X \equiv_c Y$, is defined as follows:*

$$\begin{aligned}\forall X, Y \in \mathcal{P}(\mathcal{R}),\ X \equiv_c Y \Leftrightarrow \exists Z \in \mathcal{P}(\mathcal{R}),\ X \subseteq Z,\ Y \subseteq Z \\ \text{such that } |r(X)| = |r(Y)| = |r(Z)|\end{aligned}$$

Definition 16 (closure equivalence relation). *Let $X, Y \in \mathcal{P}(\mathcal{R})$ be two sets, the closure equivalence relation over r, $X \equiv_h Y$, is defined as follows:*

$$\forall X, Y \in \mathcal{P}(\mathcal{R}),\ X \equiv_h Y \Leftrightarrow h(X) = h(Y)$$

Proposition 4. $\forall X, Y \in \mathcal{P}(\mathcal{R})$,

$$X \equiv_\pi Y \Leftrightarrow X \equiv_c Y \Leftrightarrow X \equiv_h Y$$

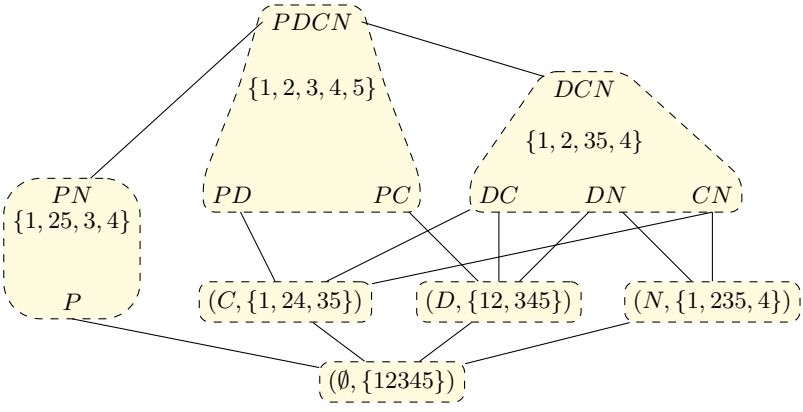


Fig. 3. Hasse diagram of the Quotient Agree Lattice of the relation HOUSING

Proof.

1. $X \equiv_h Y \Leftrightarrow h(X) = h(Y)$. Moreover, according to proposition 2, $\pi_X = \pi_{h(X)}$ and $\pi_Y = \pi_{h(Y)}$, since $h(X) = h(Y)$ we have $\pi_X = \pi_Y$. □
2. According to (Stumme et al., 2002) counting is a weight function hence $X \equiv_h Y \Leftrightarrow X \equiv_c Y$. □

The above proposition states the relationship between the Quotient Agree Lattice and the concepts related to the Agree Concept Lattice.

5 Applications to Multidimensional Database Analysis

In this paper we have proposed two soundly founded structures originated from formal concept analysis and database theory. In this section, we highlight three database applications for which these structures are of great interest. The former two result from classical issues: mining exact and approximate functional dependencies and computing data cubes. The latter approach provides a multidimensional and multi-criterion database analysis intended for retrieving the user preferences. It is called SKYCUBE.

5.1 Functional Dependencies

The discovery of functional dependencies is a well known task in database design and tuning (Beeri et al., 1984; Lopes et al., 2002; Novelli and Cicchetti, 2001). All the approaches tackling this issue mine a canonical cover which encompasses all the minimal functional dependencies. However such a cover does not exist for approximate dependencies. The concepts of agree set and partition have been successfully used for solving the dependency extraction problem with the DEPMINER approach (Lopes et al., 2000). With the Quotient Agree Lattice, it is

possible to easily compute both approximate and exact functional dependencies. All the dependencies within classes are exact dependencies. They hold between the minimal generators and associated closed set. Dependencies between classes are approximate ones. They link between minimal generators and closed sets (Bastide et al., 2000; Pasquier et al., 2005; Stumme et al., 2002).

Finally the covering graph of the Quotient Agree Lattice is a visual tool which makes it possible to easily detect the various functional dependencies.

Example 10. With the Quotient Agree Lattice of the relation HOUSING (Cf. figure 3), the exact functional dependency $DC \rightarrow N$ holds because DC and DCN belong to the same equivalence class. $N \rightarrow DC$ is an approximate dependency because N is a minimal generator and DCN is a closed set of a different but linked class.

5.2 Data Cubes

A data cube (Gray et al., 1997; Morfonios et al., 2007) stores all the aggregates according to all the possible combinations of attributes (all the possible GROUP-BY). The result of any GROUP-BY operation is called a cuboid. To compute data cubes, an approach based on database partitions has been proposed: PCUBE (Casali et al., 2009b; Laporte et al., 2002). It navigates within the power set lattice of the set of attributes from bottom to top. For each node in the lattice, it performs the product of its predecessor partitions in order to yield the associated cuboid. With the Agree Concept Lattice, we have shown that for each node X in the lattice we have: $\pi_C = \pi_{h(C)}$. By applying this property, it is possible to avoid the computation of the partition product for the non-closed attribute sets. Furthermore it is not necessary to store the whole data cube, only the cuboids associated with the Agree Concepts.

Example 11. With our relation, the 2^4 partitions computed by PCUBE are: $\pi_\emptyset$, π_P, π_D, π_C, π_N, π_{PD}, π_{PC}, π_{PN}, π_{DC}, π_{DN}, π_{CN}, π_{PDC}, π_{PCN}, π_{PDN}, π_{DCN}, π_{PDCN}. By using the Agree Concept Lattice, they are reduced to: $\pi_\emptyset$, π_D, π_C, π_N, π_{PN}, π_{DCN}, π_{PDCN}. In this small example, with the Agree Concept Lattice we avoid the computation and storage of 9 partitions.

5.3 Skycubes

The SKYLINE operator (Börzsönyi et al., 2001) is intended to retrieve the most relevant objects: the tuples which are not dominated by any other tuple (Pareto's dominance). It is originated by the *maximal vector problem* (Kung et al., 1975).

The SKYLINE operator is a decision making tool and the user will likely compute several SKYLINES before finding the ones which are really interesting. In order to address such an issue, the SKYCUBE (Yuan et al., 2005; Pei et al., 2005) has been proposed. The underlying general idea is to compute all the possible SKYLINES of all subspaces (subsets of attributes) beforehand, then it is necessary to reduce as much as possible the storage cost of the result.

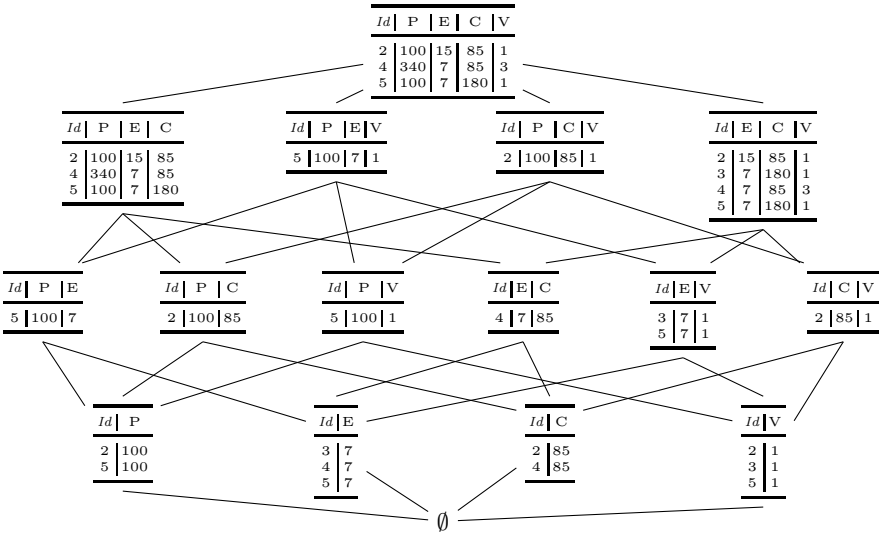


Fig. 4. Representation as a lattice of the SKYCUBE of the relation HOUSING

According to Pei et al. (2005), if a tuple t belongs to the SKYLINE of the subspaces C_1 and C_2 such that $C_1 \subset C_2$, t does not necessarily belong to the SKYLINE of any subspace C located between C_1 and C_2 ($C_1 \subset C \subset C_2$). Such a property would be especially interesting since it could significantly reduce the multidimensional SKYLINE computation. Unfortunately it does not hold in the general case: belonging to a SKYLINE is not monotonic.

Like for the data cube, the SKYCUBE may enclose superfluous information. This feature has motivated the proposal of reduced representations of the SKYCUBE presented in Pei et al. (2006). In order to avoid the important cost of the reconstruction of SKYLINES originated from the value oriented representation of Pei et al. (2006), the Agree Concept Lattice can be used as a basis for an attribute oriented reduction method.

Using the Agree Concepts to compute the SKYCUBE has several advantages:

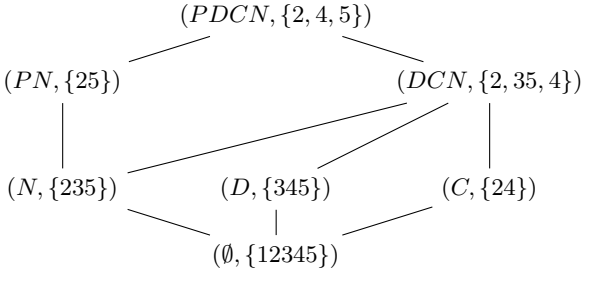


Fig. 5. Partial materialization of the SKYCUBE based on the Agree Concept Lattice for the relation HOUSING

1. If a tuple t belongs to the SKYLINE of the subspace C, all the tuples $t' \in [t]_C$ also belong to this SKYLINE. Therefore, by using partitions in order to compute a SKYLINE, we avoid a great number of useless comparisons.
2. According to proposition 2, $\pi_C = \pi_{h(C)}$ hence we only need the Agree Concepts to efficiently compute the SKYCUBE.
3. In Nedjar et al. (2011), we have shown that the SKYLINE according to C is included in the SKYLINE according to $h(C)$. The Agree Concept lattice is the first partial materialization which is attribute oriented. Such a materialization makes it possible to efficiently compute the missing cuboids from the SKYLINE objects associated with their Agree Concepts.

The objective of this approach is the reconstruction at the least cost of the Skycuboids, and it ideally behaves when such a task must be performed. Despite its targeted orientation, it is a good compromise between data actualization and storage space reduction.

Example 12. With our relation HOUSING, the Figure 4 gives the associated SKYCUBE and Figure 5 gives its partial materialization based on the Agree Concept Lattice.

6 Conclusion

In this paper, we have proposed two lattice structures at the cross road between formal concept analysis and database theory: the Agree Concept Lattice and the Quotient Agree Lattice. One of the main features of our structures is their generic feature and they can apply in several application fields, in particular for partially materializing SKYCUBES. Our aim when proposing these structures is to make use of database concepts to solve database and OLAP problems. This makes it possible to take advantage of existing and efficient database tools. We are currently working on an algorithmic approach, intended to be integrated within DBMSs and which takes advantage of our structure's adaptability. The final objective is to achieve a unified software platform devoted to multidimensional and multicriterion database analysis.

References

Abiteboul, S., Hull, R., Vianu, V.: Foundations of databases, vol. 8. Citeseer (1995)

Barbut, M.: Partitions d'un ensemble fini: leur treillis (cosimplexe) et leur représentation géométrique. Mathématiques et Sciences Humaines 22, 5–22 (1968)

Bastide, Y., Pasquier, N., Taouil, R., Stumme, G., Lakhal, L.: Mining minimal non-redundant association rules using frequent closed itemsets. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 972–986. Springer, Heidelberg (2000)

Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of armstrong relations for functional dependencies. J. ACM 31(1), 30–46 (1984)

Birkhoff, G.: Lattice Theory, 3rd (new) edn. AMS Colloquium Publications, vol. XXV. American Mathematical Society, Providence (1970)

Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430. IEEE Computer Society, Los Alamitos (2001)

Casali, A., Cicchetti, R., Lakhal, L.: Extracting semantics from data cubes using cube transversals and closures. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) KDD, pp. 69–78. ACM, New York (2003)

Casali, A., Nedjar, S., Cicchetti, R., Lakhal, L.: Closed cube lattices. Annals of Information Systems 3(1), 145–164 (2009a); New Trends in Data Warehousing and Data Analysis

Casali, A., Nedjar, S., Cicchetti, R., Lakhal, L., Novelli, N.: Lossless reduction of datacubes using partitions. IJDWM 5(1), 18–35 (2009)

Diallo, T., Novelli, N., Petit, J.-M.: Discovering (frequent) constant conditional functional dependencies. The International Journal of Data Mining, Modelling and Management, IJDMMM (2011)

Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. Data Min. Knowl. Discov. 1(1), 29–53 (1997)

Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. J. ACM 22(4), 469–476 (1975)

Lakshmanan, L.V.S., Pei, J., Han, J.: Quotient cube: How to summarize the semantics of a data cube. In: Lochovsky, F.H., Shan, W. (eds.) VLDB, pp. 778–789. Morgan Kaufmann, San Francisco (2002)

Laporte, M., Novelli, N., Cicchetti, R., Lakhal, L.: Computing full and iceberg datacubes using partitions. In: Hacid, M.-S., Raś, Z.W., Zighed, D.A., Kodratoff, Y. (eds.) ISMIS 2002. LNCS (LNAI), vol. 2366, pp. 244–254. Springer, Heidelberg (2002)

Lopes, S., Petit, J.-M., Lakhal, L.: Efficient discovery of functional dependencies and armstrong relations. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 350–364. Springer, Heidelberg (2000)

Lopes, S., Petit, J.-M., Lakhal, L.: Functional and approximate dependency mining: database and fca points of view. J. Exp. Theor. Artif. Intell. 14(2-3), 93–114 (2002)

Medina, R., Nourine, L.: Conditional functional dependencies: An FCA point of view. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA 2010. LNCS, vol. 5986, pp. 161–176. Springer, Heidelberg (2010)

Morfonios, K., Konakas, S., Ioannidis, Y.E., Kotsis, N.: Rolap implementations of the data cube. ACM Comput. Surv. 39(4) (2007)

Nedjar, S., Casali, A., Cicchetti, R., Lakhal, L.: Emerging cubes: Borders, size estimations and lossless reductions. Information Systems 34(6), 536–550 (2009)

Nedjar, S., Casali, A., Cicchetti, R., Lakhal, L.: Constrained closed datacubes. In: Kwuida, L., Sertkaya, B. (eds.) ICFCA 2010. LNCS, vol. 5986, pp. 177–192. Springer, Heidelberg (2010a)

Nedjar, S., Casali, A., Cicchetti, R., Lakhal, L.: Reduced representations of emerging cubes for olap database mining. IJBIDM 5(1), 268–300 (2010)

Nedjar, S., Cicchetti, R., Lakhal, L.: Constrained Closed and Quotient Cubes, ch. 5. SCI, vol. 2. Springer, Heidelberg (2010)

Nedjar, S., Pesci, F., Cicchetti, R., Lakhal, L.: Treillis des concepts skylines: Analyse multidimensionnelle des skylines fondée sur les ensembles en accord. In: EGC 2011 – Extraction et Gestion Des Connaissances, Revue des Nouvelles Technologies de l'Information, Cépaduès-Éditions (2011)

Novelli, N., Cicchetti, R.: Functional and embedded dependency inference: a data mining point of view. Inf. Syst. 26(7), 477–506 (2001)

Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Information Systems 24(1), 25–46 (1999)

Pasquier, N., Taouil, R., Bastide, Y., Stumme, G., Lakhal, L.: Generating a condensed representation for association rules. J. Intell. Inf. Syst. 24(1), 29–60 (2005)

Pei, J., Jin, W., Ester, M., Tao, Y.: Catching the best views of skyline: A semantic approach based on decisive subspaces. In: VLDB, pp. 253–264 (2005)

Pei, J., Yuan, Y., Lin, X., Jin, W., Ester, M., Liu, Q., Wang, W., Tao, Y., Yu, J.X., Zhang, Q.: Towards multidimensional subspace skyline analysis. ACM Trans. Database Syst. 31(4), 1335–1381 (2006)

Spyratos, N.: The partition model: A deductive database model. ACM Trans. Database Syst. 12(1), 1–37 (1987)

Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with titanic. Data Knowl. Eng. 42(2), 189–222 (2002)

Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J.X., Zhang, Q.: Efficient computation of the skyline cube. In: VLDB, pp. 241–252 (2005)

Zaki, M.J.: Generating non-redundant association rules. In: KDD, pp. 34–43 (2000)

Abstract Concept Lattices

Henry Soldano[1] and Véronique Ventos[2]

[1] L.I.P.N, UMR-CNRS 7030, Université Paris-Nord,
93430 Villetaneuse, France
[2] LRI, UMR-CNRS 8623, Université Paris-Sud,
91405 Orsay, France
henry.soldano@lipn.univ-paris13.fr, ventos@lri.fr

Abstract. We present a view of abstraction based on a structure preserving reduction of the Galois connection between a language $\mathcal{L}$ of terms and the powerset of a set of instances O. Such a relation is materialized as an *extension-intension lattice*, namely a concept lattice when L is the powerset of a set P of attributes. We define and characterize an *abstraction* A as some part of either the language or the powerset of O, defined in such a way that the extension-intension latticial structure is preserved. Such a structure is denoted for short as an *abstract lattice*. We discuss the extensional abstract lattices obtained by so reducing the powerset of O, together together with the corresponding abstract implications, and discuss *alpha lattices* as particular abstract lattices. Finally we give formal framework allowing to define a *generalized abstract lattice* whose language is made of terms mixing abstract and non abstract conjunctions of properties.

1 Introduction

There were in machine learning various attempts to formalize abstraction and characterize its desirable properties with respect to induction. An important statement was that abstraction should be order-preserving with respect to the partially ordered language in which hypotheses are searched for [21]. However, the question of what classes of abstractions are to be investigated for learning and reasoning is far from being exhausted. We present a view of abstraction based on a structure preserving reduction of the relation between a term t of a language $\mathcal{L}$, partially ordered following a general-to-specific partial order, and the *extension* of t on a set of objects (or instances) O, representing the subset of O whose elements satisfy the term.

In Formal Concept Analysis [15] and Galois lattice theory [4] $\mathcal{L}$ is a lattice, and the relation between $\mathcal{L}$ and the powerset $\mathcal{P}(O)$ is materialized as an *extension-intension lattice*. This lattice is the structure of the *definable* elements of $\mathcal{P}(O)$ [1], *i.e.* the subsets of O that are each the extension of some term of $\mathcal{L}$. In such a lattice, a definable set e represents the equivalence class of all the terms whose extension is e, and a node, also called a *concept*, is a pair (e, t) where t is the most specific term of the class, denoted as the *intension* of the concept. Formal Concept Analysis is primarily concerned with the relation between the powerset of a set of properties as a language and $\mathcal{P}(O)$. The extension-intension lattice is then denoted as a *concept lattice*. Various extensions have been recently proposed to ease the representation in more sophisticated languages

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 235–250, 2011.

[6,18]. In particular *pattern structures* [16] have been recently introduced to represent complex data, associating such a *pattern structure* to each object. Logical Concept Analysis (LCA) [13] has been recently introduced as a general formalization in which $\mathcal{L}$ is a logical language and uses *object descriptions* in $\mathcal{L}$. Though we do not use here the LCA formulation and notations, for technical reasons, our construction of a Galois connection on a modal language is very similar to the construction presented in [12]. In a recent paper [20] particular mappings, denoted as *projections*, are used to reduce $\mathcal{L}$ or $\mathcal{P}(O)$ in such a way that the relation between the language and the extensional space is still materialized as a lattice. In other words, projections ensure that we have a coarser, yet structure preserving, view of concepts representatives of the universe we deal with. Independently [16] also uses projections on pattern structures.

In this paper we show that projections of a lattice are in a one to one correspondence with *abstractions* defined as parts of the lattice that are closed under least upper bound. We call the corresponding structures *abstract extension-intension lattices* and *abstract concept lattice*, and for short *abstract lattices*. We first briefly discuss intensional abstractions and investigate then more specifically extensional abstractions, i.e. parts of $\mathcal{P}(O)$ that are closed under set theoretic union.

More precisely, applying an extensional abstraction means that we will no longer consider instances of O as elements of the extensional space, but rather consider subsets of O given *a priori*. As an example consider $O = \{o_1, o_2, o_3, o_4\}$, and A be obtained by closing under union the part $\{\{o_1, o_2\}, \{o_1, o_3\}\}$. As a result, $\{o_1, o_2, o_3\}$ belongs to A but $\{o_2, o_3\}$ does not. Now, consider the smallest elements of A that contain a given instance o. We call these elements the *minimal abstractions* of o and consider them as *abstract instances*. In this example, $\{o_1, o_2\}$ is the unique minimal abstraction of o_2, $\{o_1, o_3\}$ is the unique minimal abstraction of o_3, $\{o_1, o_2\}$ and $\{o_1, o_3\}$ are the two minimal abstractions of o_1, and o_4 has no abstraction in A.

We relate then any term t to an *abstract extension* $ext_A(t)$ that turns out to be the union of all the abstract instances included in $ext(t)$. Going back to our example, suppose that $ext(t) = \{o_1, o_2, o_3, o_4\}$, then the abstract instances included in $ext(t)$ are $\{o_1, o_2\}$ and $\{o_1, o_3\}$ and therefore $ext_A(t) = \{o_1, o_2, o_3\}$. Clearly the abstract extension of a given term is always included in its original extension. In other words we reduce the extension of the term by excluding any instance that has no minimal abstraction included in the original extension. In the current example, o_4 is such an instance. The intuition here is a change in granularity: the new objects we deal with are the minimal abstractions of the original instances. Furthermore, as an extension-intension lattice represents a set of valid implications (see for instance [4]), our *abstract extension-intension lattice* also represents a set of *valid abstract implications*. Going back to our example, suppose $ext(p) = \{o_1, o_2, o_4\}$ and $ext(q) = \{o_1, o_2, o_3\}$, then the implication $p \rightarrow q$ is not valid on O. However, as we have $ext_A(p) = \{o_1, o_2\}$ and $ext_A(q) = \{o_1, o_2, o_3\}$, the abstract implication $p \rightarrow_A q$ is valid. As a matter of fact a general property of extensional abstractions is that they preserve validity of implications. Algorithms that extract valid implications [14,17] from a set of instances O, can be extended to extract *valid abstract implications*. As an example, we can interpret *alpha lattices* [28] as particular abstract lattices, and alpha implications are straightforwardly extracted by extending the method of N. Pasquier and collaborators [19] to alpha lattices.

Regarding extensional abstractions, the formal work closest to ours regards rough sets logics. Rough sets originally relies on an indiscernibility relation (see for instance [29]). More recently, generalization of rough sets using coverings, corresponding to our extensional abstractions, has been investigated from an algebraic perspective [5,8]. However these works do not investigate the extension-intension relationship.

To summarize, in this first part of the paper we characterize and discuss the properties of abstractions, and particularly of extensional abstractions, as a structure and order preserving reduction of extension-intension lattices.

We then remark that abstract implications are in fact implications between abstract terms, *i.e.* $t_1 \rightarrow t_2$ rewrites as $\Box t_1 \rightarrow \Box t_2$, where $\Box t$ means "*Abstractly* t" with respect to the abstraction A. Then, starting from the semantics, we discuss the corresponding *abstract modal logics*. This leads us to consider a language $\mathcal{L}\Box$ a term of which is made of a non abstract part together with an abstract part and to define a new, more expressive, extension-intension lattice.

2 Preliminaries

2.1 Galois Connection

We recall here the definitions of a Galois connection and a Galois lattice.

Definition 1. *Given two lattices* $(E, \leq_E, \wedge, \vee)$ *and* $(F, \leq_F, \wedge, \vee)$*, where* $\leq_E$ *and* $\leq_F$ *denote the order relations, and* $\wedge, \vee$ *the meet and the join operations, a Galois connection between* E *and* F *is a pair of mappings* $(f : E \rightarrow F,\ g : F \rightarrow E)$ *verifying the following properties:*

- *For any* x *and* x' *in* E*, we have that* $x \leq_E x'$ *implies* $f(x) \geq_F f(x')$
- *For any* y *and* y' *in* F *, we have that* $y \leq_F y'$ *implies* $g(y) \geq_E g(y')$
- *For any* x *in* E *and* y *in* F *we have that* $g \circ f(x) \geq_E x$ *and* $f \circ g(y) \geq_F y$

The Galois lattice defined by the Galois connection (f, g) *is then the set* $\{(x, y) \in E \times F \mid y = f(x)$ *and* $x = g(y)\}$ *ordered by* $\leq_E$ *.*

2.2 Extension-Intension Lattices

FCA, in a broad sense, investigates the link between the terms of some language $\mathcal{L}$ and a universe O of elements of the universe denoted as *instances*. We also call *intensional* representations the terms of $\mathcal{L}$ and extensional representations the elements of the powerset $\mathcal{P}(O)$. In this presentation we suppose that we know whenever an instance o satisfies the term t, and we define accordingly its extension:

Definition 2. $ext_O(t) = \{o \in O \mid o \textit{ satisfies } t\}$.

In what follows we consider O as a fixed and finite set of instances and we simply write $ext(t)$ the extension in O.

The language $\mathcal{L}$ is partially ordered by a general-to-specific relation. We write $t1 \preceq t2$ whenever t_1 is *less specific* than t_2, or equivalently t_1 is more general than t_2. The following proposition relate $\mathcal{L}$ and $\mathcal{P}(O)$ by a Galois connection.

Proposition 1. *Let $\mathcal{L}$ be a finite language, $\preceq$ a partial order on $\mathcal{L}$ denoted as specificity, O be a finite set of instances and ext : $\mathcal{L} \rightarrow \mathcal{P}(O)$ be a mapping such that $t_1 \preceq t_2 \Rightarrow ext(t_1) \supseteq ext(t_2)$. We consider the following conditions:*

- *(condition 2.1) For any instance o, there is a* unique most specific term*, denoted as the object description $d(o)$, among all terms t such that $o \in ext(t)$*
- *(condition 2.2) $\mathcal{L}$ has a greatest element and is a lower semi lattice,* i.e. *each pair of terms t_1, t_2 of $\mathcal{L}$ has a unique greatest lower bound $t_1 \wedge_{\mathcal{L}} t_2$ in $\mathcal{L}$, also called the* least general generalisation *(for short* lgg*) of t_1 and t_2.*

Whenever conditions 2.1 and 2.2 are satisfied, $(\mathcal{L}, \preceq)$ is a lattice, i.e. *two terms t_1 and t_2 also have a least upper bound denoted $t_1 \vee_{\mathcal{L}} t_2$, and the pair (int, ext) where*

$$int(e) = \bigwedge_{o \in e_{\mathcal{L}}} d(o)$$

is a Galois connection.

In Formal Concept Analysis [15] and Galois Analysis [4], $\mathcal{L}$ is the powerset of a set P of attributes. Recent extensions to various languages have been performed [14,13]. In particular in [16], $\mathcal{L}$ is defined as a set of *pattern structures*, *i.e.* terms generated by first considering the set of object descriptions, and then closing it under the least general generalization $\wedge_{\mathcal{L}}$. Independently, E. Diday and R. Emilion start from the same assumptions [9]. Proposition 1 directly follows from, for instance, theorem 2 in [9].

We call *extension-intension lattice* the Galois lattice G coresponding to this Galois connection. G is ordered using the extensional order and each element G is a pair (e, t) such that $t = int(e)$ and $e = ext(t)$, *i.e.* t is the unique most specific term representing the equivalence class of terms whose extension is e, and is also denoted as the *intension* of e. Such most specific terms are also referred as *closed terms* or *closed motifs* in data mining [25]. In Machine Learning, the search space $\mathcal{L}$ may be in this way explored by minimally generalizing or specializing such closed terms [3].

Finally G is considered as the structure of O as perceived through $\mathcal{L}$ and can be also represented by the set T_O of all the implications $p \rightarrow q$ which are valid on O, i.e. such that $ext(p) \subseteq ext(q)$ is true. The equivalence class whose *intension* is $t = int(e)$, also contains various minimal (i.e. most general) terms t^m, also know as *generators*. The elements of T_O can be generated from the set of all the $t^m \rightarrow t$ implications also called the *min-max implications* basis of T_O [19].

2.3 Projections and Projected Extension-Intension Lattices

Now, consider the following problem; how to reduce $\mathcal{P}(O)$, $\mathcal{L}$, or both, in such a way that *i)* conditions 2.1 and 2.2 are still satisfied, *ii)* the resulting extension-intension lattice G' is isomorphic to part of G. In other words, how to reduce G by reducing the language or the powerset of the universe of instances in such a way that the structure is preserved, and its size reduced ? An answer to this question is given in [20] through the use of intensional and extensional projections whose images are also lattices:

Definition 3 (Projection). *p is a projection of a lattice $(M, \leq)$ iff for each pair (x, y) of elements of M, we have :*

If $x \leq y$ *then* $p(x) \leq p(y)$ *(monotonicity)*
$p(x) \leq x$ *(minimality)*
$p(x) \leq p(p(x))$ *(semi-idempotence)*

Let M be a lattice and p a projection of M, then $p(M)$ is also a lattice with join operator $\vee$ and meet operator $\wedge$ defined as, for any pair $m_1, m_2 \in p(M)$, $m_1 \wedge m_2 = p(m_1 \wedge_M m_2)$ and $m_1 \vee m_2 = m_1 \vee_M m_2$.

When considering respectively $M = \mathcal{L}$ and $M = \mathcal{P}(O)$ we obtain respectively intensional and extensional projections and both lead to *projected extension-intension lattices* [20,28].

Proposition 2. *Let* (int, ext) *be a Galois connection on* $(\mathcal{P}(O), \preceq), (\mathcal{L}, \subseteq))$*,* G *be the associated Galois lattice, and* (e, t) *be a node of* G*.*

- *Let* p *be a projection on* $\mathcal{L}$*, then* $(p \circ int, ext)$ *defines a Galois connection on* $((p(\mathcal{L}), \preceq), (\mathcal{P}(O), \subseteq))$ *and* (e, t) *is projected in the corresponding Galois lattice* $p(G)$ *on the node* (e', t') *such that* $t' = p(t)$ *and* $e' = ext(t')$*.*
- *Let* p *be a projection on* $\mathcal{P}(O)$*, then* $(int, p \circ ext)$ *defines a Galois connection on* $((\mathcal{L}, \preceq), (p(\mathcal{P}(O)), \subseteq))$ *and* (e, t) *is projected in the corresponding Galois lattice* $p(G)$ *on the node* (e', t') *such that* $e' = p(e)$ *and* $t' = int(e')$*.*

Note that in partial order theory, projections are known as *kernel operators* or *interior operators*. Their properties are well known [11] and are the basis of the next section.

3 Abstractions

3.1 From Projection to Abstractions

Reducing through projections $\mathcal{L}$, $\mathcal{P}(O)$ or both results in a reduced extension-intension representation whose latticial structure is preserved. However, while projections are technically useful, they do not always give a simple way to chose simplified representations. In what follows we present an equivalent view of projections.

Definition 4. *An abstraction of a lattice* M *is a subset*[1] *of* M*, closed under* $\vee_M$*.*

Building abstractions therefore simply means to chose any subset of M and close it by $\vee_M$. We note hereunder that abstractions are in a one to one correspondence with projections and so *abstract extension-intension lattices* are defined as projected extension-intension lattices.

Proposition 3. *Let* A *be an abstraction of a lattice* M*, then* p_A *defined as* $p_A(x) = \bigvee_{c \in A, c \leq x} c$*, is a projection of* M*. Let* p *be a projection then* $A = p(M)$ *is an abstraction of* M*, and* p *is the projection* p_A *associated to* A*.*

The equivalence between *interior systems* $p(M)$ and subsets A of M closed under union is a known property of interior operators [11]. As abstractions are closed under $\vee_M$, we have that $\vee_M = \vee_A$. From now on we simply write $\vee$ when no confusion is possible. An important point is that we only need the $\vee$-irreducible elements[2] of A:

[1] Note that $\perp_M$, the smallest element of M, belongs to all abstractions.
[2] Irreducible elements of A are elements that cannot be obtained as a result of applying $\vee_M$.

Proposition 4. *Let A be an abstraction of M, and A_I be the set of $\vee$-irreducible elements of A, the projection p_A is obtained by only considering elements of A_I:*

$$p_A(x) = \bigvee_{c \in A_I, c \leq x} c$$

Intensional abstractions An *intensional abstraction* is then simply obtained by selecting part of the language $\mathcal{L}$ and closing it by the join operator $\vee_{\mathcal{L}}$. For instance, consider the lattice $\mathcal{L}$ whose elements are intervals $[a, b]$ such that $a, b \in \{1, 2, 3, 4\}$. We have then $[1, 3] \vee_{\mathcal{L}} [3, 4] = [3]$. Consider first the abstraction A whose elements are the intervals containing 3. A is closed by $\vee_{\mathcal{L}}$ as intersecting two intervals containing 3 results in an interval also containing 3.

Consider now $\mathcal{L}' = \mathcal{L} - \{[1], [2], [3], [4]\}$. $\mathcal{L}'$ is obtained by simply deleting the most specific terms and is clearly also a lattice. However $[1, 3] \vee_{\mathcal{L}'} [3, 4]$ is now [] and so clearly $\mathcal{L}'$ is not closed under $\vee_{\mathcal{L}}$, and therefore is not an abstraction. Note that there are in $\mathcal{L}'$ two most specific elements satisfied by $o = 3$, and as a consequence $d(o)$ is no longer defined.

Extensional abstractions. They abstract instances rather than abstracting the language of terms. Note that when considering extensional abstractions, $\vee$ is the set theoretic union $\cup$, and that the order relation is the set theoretic inclusion.

Example 1. As an example consider $O = \{1, 2, 3, 4\}$, and the abstraction A obtained by closing under union the part $\{\{1, 2\}, \{1, 3\}\}$ of $\mathcal{P}(O)$, so adding $\{1, 2, 3\}$and $\emptyset$ to build A. The set of $\cup$-irreducible elements of A is $A_I = \{\{1, 2\}, \{1, 3\}\}$. For instance, $p_A(\{1, 2, 3\}) = \{1, 2\} \cup \{1, 3\} = \{1, 2, 3\}$ as both elements of A_I are included in $\{1, 2, 3\}$, and $p_A(\{2, 3\}) = \emptyset$ because neither $\{1, 2\}$ nor $\{1, 3\}$ is included in $\{2, 3\}$.

We remark in the next section that abstractions are partially ordered.

3.2 The Lattice of Abstractions

There is a partial order on projections of a lattice M [20] and therefore, on abstractions:

Definition 5. *Let M be a lattice and p_1 et p_2 two projections of M, we will state that $p_2 \leq p_1$, i.e. p_2 is less* concrete*, and so more* abstract *than p_1, iff there is some projection p defined on $p_1(M)$ such that for all c in M, $p_2(c) = p \circ p_1(c)$.*

This means that $A_2 = p_2(M)$ is more abstract than $A_1 = p_1(M)$ *iff* A_2 is an abstraction of A_1. This is also equivalent to saying that any $\vee$-irréductible element i_2 of A_2 may be written as a disjunction of $\vee$-irreducible elements of A_1, *i.e.* $i_2 = i_1^1 \vee \cdots \vee i_1^n$

Example 2 (Extensional abstractions). Let $M = \mathcal{P}(\{1, 2, 3, 4\})$, $A_{I1} = \{\{1, 2\}, \{1, 3\}\}$ and $A_{2I} = \{\{1, 2\}, \{1, 2, 3\}\}$, A_2 is more abstract than A_1 as $\{1, 2\} \in A_{I1}$ and $\{1, 2, 3\} = \{1, 2\} \cup \{1, 3\}$.

Proposition 5 (Lattice of abstractions). *Let M be a lattice, and $\mathcal{A}$ be the set of the abstractions of M. Let then $A_1 = p_1(M)$ and $A_2 = p_2(M)$ be elements of $\mathcal{A}$:*

- *The least upper bound* $A_1 \vee_{\mathcal{A}} A_2$ *is obtained by closing* $A_1 \cup A_2$ *under* $\vee$.
- *The greatest lower bound* $A_1 \wedge_{\mathcal{A}} A_2$ *is simply* $A_1 \cap A_2$.

This partial order, defined through projections, is known to order projected extension-intension lattices [20]. Therefore, abstract extension-intension lattices, where abstraction is performed on $\mathcal{L}$ or $\mathcal{P}(O)$, are also ordered following the lattice of abstractions.

4 Extensional Abstractions and Extensional Abstract Lattices

4.1 Extensional Abstract Lattices

An extensional abstraction is obtained by considering part of the powerset of O, and closing it by the union operator $\cup$. This means that we do not consider any more instances but rather subsets of instances, called *abstract instances*.

Definition 6. *Let* A *be an extensional abstraction and* p_A *the associated projection of* $\mathcal{P}(O)$*, then let* t *be a term of* $\mathcal{L}$*,* $ext_A(t) = p_A \circ ext(t)$ *is the* abstract extension *of* t*.*

From Proposition 4 we also have

$$ext_A(t) = \bigcup_{u \in A_I, u \subseteq ext(t)} u$$

Definition 7. *Let* A *be an abstraction of* $\mathcal{P}(O)$*, the projected extension-intension lattice* $p_A(G)$ *is denoted as an extensional abstract lattice, and noted* G_A*.*

Hereunder we define *abstract instances* and *minimal abstractions* of a given instance.

Definition 8. *Let* A *be an extensional abstraction.*

- *An element of* A_I *is called an* abstract instance.
- *Let* $A_I(o)$ *be the subset of* A_I *whose elements contain the instance* o*. We denote as minimal abstractions of an instance* o *the minimal elements of* $A_I(o)$*, i.e.* $A_m(o) = \{u \in A_I \mid o \in u \text{ and if } o \in u' \subset u, \text{ then } u' \notin A\}$.

An interesting point is that A_I is the set of minimal abstractions of the instances:

Proposition 6.

$$A_I = \bigcup_{\{o \in O\}} A_m(o)$$

Example 3. In this example $O = \{1, 2, 3, 4\}$ and $A_I = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\}$. Here $\mathcal{L} = \mathcal{P}(P)$ where $P = \{X, Y, Z\}$ is a set of properties. We give hereunder a table representing the context relating $\mathcal{P}(P)$ to $\mathcal{P}(O)$. We have added a column for each element of A_I. In Figure 1 we have represented the original concept lattice together with the extensional abstract lattice (here an *abstract concept lattice*) associated to A_I.

Instances	X	Y	Z	{1,2}	{2,3}	{3,4}
1	1	0	1	1	0	0
2	1	1	1	1	1	0
3	0	1	0	0	1	1
4	1	0	0	0	0	1

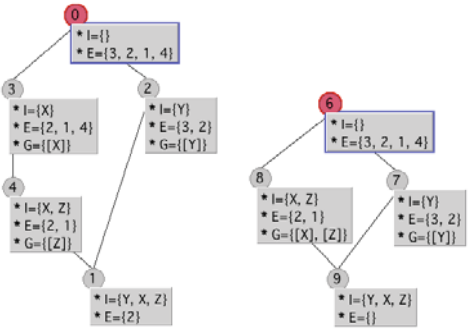


Fig. 1. The concept lattice (on the left) and the abstract concept lattice (on the right) corresponding to the context and abstraction given in Example 3. On each node I is the intension, E is the extension, and G is the set of most general terms whose extension is E. The nodes 3 and 4 of the original concept lattice are merged into the node 8 of the abstract concept lattice. As a result, the abstract implication $X \rightarrow Z$ is now valid as X and Z have the same abstract extension.

We may note that such an extensional abstract lattice is isomorphic to an extension-intension lattice relating $\mathcal{L}$ to the powerset of A_I. For that purpose, we reformulate A_I as a new instance set and we define $(ext' : \mathcal{L} \rightarrow A_I)$ as follows: let $u = \{o_1, \ldots o_k\}$ be an element of A_I, we have then that u, as a new instance, belongs to $ext'(t)$ whenever u, as a subset of O, is included in $ext(t)$. As a consequence $ext_A(t) = \bigcup_{u \in ext'(t)} u$ and therefore the images $ext'(\mathcal{L})$ and $ext_A(\mathcal{L})$ are in one to one correspondence.

As stated before, an extension-intension lattice corresponds to a set of valid implications. We define hereunder the abstract implications associated to an extensional abstract lattice.

Definition 9 (Abstract implication). *Let t_1 et t_2 be terms of $\mathcal{L}$. Whenever $ext_A(t_1) \subseteq ext_A(t_2)$ we say that the abstract implication $t_1 \rightarrow_A t_2$ is valid on A.*

Then G_A is represented by the whole set of abstract implications valid on A, or by any generating subset, as the *min-max basis of abstract implications* extending to the extensional abstract lattices the definition of the min-max basis of implications (see end of Section 2.2).

Now as $t_1 \rightarrow t_2$ means that $ext(t_1) \subseteq ext(t_2)$, we have that whenever some abstract instance u is included in $ext(t_1)$, we also have that u is included in $ext(t_2)$. This also means, by definition, that $ext_A(t_1)$ is included in $ext_A(t_2)$. As a consequence validity of implications is preserved by extensional abstraction:

Proposition 7. *Let A be an abstraction of $\mathcal{P}(O)$, and t_1 and t_2 two terms of $\mathcal{L}$. If $t_1 \rightarrow t_2$ is valid on O, then the abstract implication $t_1 \rightarrow_A t_2$ is valid on A.*

4.2 Alpha Lattices as Extensional Abstract Lattices

The partial order of abstractions (see Definition 5), means that A' is more abstract than A whenever any element of A' (or A'_I) may be written as the union of elements of A

(or A_I). An interesting case is *alpha abstraction*. Starting from a subset $\mathcal{C}$ whose union closure is an initial abstraction A, more concrete abstractions A^α are built. A^α is obtained by deriving from each initial category C the set of frequent enough parts of C, and closing under union the resulting set $\mathcal{C}^\alpha$. We have then that A^α is more abstract than $A^{\alpha'}$ iff $\alpha \geq \alpha'$. In a previous work the corresponding *alpha Galois lattices* were defined through projections and experimented in order to extract alpha association rules [28]. As an example, consider the implication "Animals that fly are oviparous", it is not valid on O because of the *bat*. When considering the categorization $\mathcal{C} = \{mammal, insect, bird\}$, the corresponding abstract implication is valid but never applies: none of the initial categories contains only flying animals. However by considering frequent enough parts of each category, the corresponding $(\alpha = 0.1)$- implication is valid, as the bat is eliminated from the premise (very few mammals fly), but still applies to flying birds and flying insects, as they represent large enough parts of their categories.

5 The Extended Abstract Lattice $G_\Box$

We will formalize the nature of abstract implications by interpreting them as classical implications relating modalized terms. So we rewrite $t_1 \rightarrow_A t_2$ as $\Box_A t_1 \rightarrow \Box_A t_2$. We first define a modal logics of abstraction built on a propositional modal language $\mathcal{L}_{mod}$. Then we restrict $\mathcal{L}_{mod}$ to a language $\mathcal{L}\Box$ using only conjunction and non nested modal connectors. Finally we present an extended abstract lattice $G_\Box$ where intensions have both modalized and non modalized parts.

5.1 Modal Logics of Abstraction

Hereunder we discuss the modal logics of abstractions. Note that in this section we denote as *worlds* the elements of O.

A modal logic, in its simplest form, is a propositional logic to which is added at least one unary modal connector $\Box$, referred to as a *necessity* operator. *Classical modal logics* are the modal logics in which formulas are given truth values through *neighborhood semantics*, also known as *minimal models semantics* [7]. This wide class of modal logics includes in particular *normal modal logics* relying on Kripke possible world semantics. We will define hereunder *abstract modal logics* as particular classical modal logics. We hereunder very informally summarize neighborhood semantics in order to relate such semantics to our extensional abstractions.

The modal language $\mathcal{L}_{mod}$ is obtained by adding the modal connector $\Box$ to a propositional language $\mathcal{L}$ built on a set of atomic propositions. So for instance, $\phi = a \wedge \Box(b \wedge c)$ belongs to $\mathcal{L}_{mod}$. In order to give a truth value to a modal formula we first consider a set of worlds O, together with a *valuation* function ext, relating each atomic proposition p to the set of worlds in which p is true. For any formula ϕ without modal connectors, the computation of $ext(\phi)$ is then straightforward, for instance $ext(a \wedge b) = ext(a) \cap ext(b)$ and represents the worlds o in which ϕ is true, or in other words, the worlds that satisfies ϕ. In order to extend ext to modal formulas we need a neighboring function

$\mathcal{N}$ relating each world o to a set of subsets of O and we say that the world o is in $ext(\Box\phi)$ whenever $ext(\phi)$ belongs to $\mathcal{N}(o)$. It is known that neighboring functions $\mathcal{N}$ are in one to one correspondence with mappings $m : \mathcal{P}(O) \rightarrow \mathcal{P}(O)$ such that $m(e) = \{o \in O \mid e \in \mathcal{N}(o)\}$. To summarize we have now that

$$ext(\Box\phi) = m \circ ext(\phi)$$

Now recall that we defined abstract extension as $p \circ ext$ where p is a projection. We will so naturally define *abstract modal logics* as classical modal logics in which the mapping m is a projection. To characterize abstract modal logics we have just to translate the properties of projection as axioms and inference rules. Detailed results on modal logics of abstraction, including multimodal logics allowing to access to various levels of abstraction, are outside the scope of this paper [22]. Now, considering the abstract extension of a term as the extension of an abstract term leads to define extension-intension lattices built on languages whose terms contain both non abstract and abstract parts. This is the subject of the remaining of this section.

5.2 Mixing Abstract and Non Abstract Statements: The Language $\mathcal{L}\Box$

We consider here a language $\mathcal{L}\Box$ where classical properties and modalized properties appear simultaneously. Technically $\mathcal{L}\Box$ is a subset of $\mathcal{L}_{mod}$ whose terms contain $\wedge$ and $\Box$ as connectors, and in which the nesting of $\Box$ is not allowed. For instance, $(a \wedge b) \wedge \Box(b \wedge c) \wedge \Box(c \wedge d) \in \mathcal{L}\Box$, but $\Box((a \wedge b) \wedge \Box(b \wedge c)) \notin \mathcal{L}\Box$.

We first give an inductive definition of $\mathcal{L}\Box$.

Definition 10 (Inductive definition of $\mathcal{L}_P$ and $\mathcal{L}\Box$). *The inductive definition of $\mathcal{L}\Box$ according to P, a non-empty set of atoms, is the following (note that we define and use the language $\mathcal{L}_P$ corresponding to terms without any $\Box$ connective):*

- $\forall a \in P$, $a \in \mathcal{L}_P$
- *Constants* $\top$ *and* $\bot \in \mathcal{L}_P$
- *if F1 and F2* $\in \mathcal{L}_P$ *then* $(F1 \wedge F2) \in \mathcal{L}_P$
- *if* $F \in \mathcal{L}_P$ *then* $F \in \mathcal{L}\Box$
- *if* $F \in \mathcal{L}_P$ *then* $\Box F \in \mathcal{L}\Box$
- *if* $F1 \wedge F2 \in \mathcal{L}\Box$ *then* $(F1 \wedge F2) \in \mathcal{L}\Box$

An intensional semantics for $\mathcal{L}\Box$

We define an intensional semantics for $\mathcal{L}\Box$ based on an algebraic approach similar to [27], [10] but for description logics. The definition is made in two steps. In the first step, an equational system which highlights the main properties of the $\mathcal{L}\Box$ connectives is given. During the second step, an homomorphism based on the equational system is defined. This homomorphism is used to map terms of $\mathcal{L}\Box$ to their *structural normal form* (snf) in the intensional semantics.

Definition 11 . (The equational system $EQ_\Box$)
$\forall$ *F1, F2, F3* $\in \mathcal{L}\Box$, $\forall$ *F, F'* $\in \mathcal{L}_P$ *:*

1. $(F1 \wedge F2) \wedge F3 = F1 \wedge (F2 \wedge F3)$
2. $F1 \wedge F2 = F2 \wedge F1$
3. $F1 \wedge F1 = F1$
4. $\top \wedge F1 = F1$
5. $\bot \wedge F1 = \bot$
6. $\Box F = \Box F \wedge F$
7. $\Box(F \wedge F') = \Box(F \wedge F') \wedge \Box F \wedge \Box F'$

The equational system $EQ_\Box$ fixes the main properties of the connectives and can be used to define an equivalence relation between terms of $\mathcal{L}\Box$. Equality modulo axioms of $EQ_\Box$ is denoted by $\equiv_{EQ_\Box}$. We use it to formalize the subsumption relation in $\mathcal{L}\Box$.

Definition 12 (Subsumption in $\mathcal{L}\Box$). *Let F1, and F2 be two terms of* $\mathcal{L}\Box$*, F1* $\preceq$ *F2 (i.e. F1 subsumes or is less specific than F2) iff* $F1 \wedge F2 \equiv_{EQ_\Box} F2$.

$EQ_\Box$ induces a class of algebras. From this class, a structural algebra can be proposed, which provides $\mathcal{L}\Box$ with an intensional semantics called $\mathcal{CL}\Box$. The elements of $\mathcal{CL}\Box$ are structures whose definition is given below. These elements are *structural normal forms* of terms of $\mathcal{L}\Box$ allowing us to obtain an unique class representative for each equivalence classes of terms. $\mathcal{CL}\Box$ can be viewed as a normalized subset of $\mathcal{L}\Box$ where the lgg is unique.

The intensional semantics $\mathcal{CL}\Box$
An element of $\mathcal{CL}\Box$ corresponding to a term T of $\mathcal{L}\Box$ denoted *snf*(T) is a pair defined as follows: $< Eclassical, E\Box >$. *Eclassical* is a set of atoms belonging to P , $E\Box$ is a set of subsets of P. Intuitively, *Eclassical* contains every explicit and implicit classical properties of T (i.e. properties not at reach to a $\Box$ connective). The data structure used can then be a simple set of atoms. $E\Box$ contains $\Box$ properties of T. Since for instance $\Box(a \wedge b)$ and $\Box(b \wedge d)$ cannot be compared, we must keep both of them. The data structure used is a set of sets.

This definition presents the data structure of elements of $\mathcal{CL}\Box$ but not how to associate to a term of $\mathcal{L}\Box$ its corresponding element in $\mathcal{CL}\Box$. To make this computation and then to define $\mathcal{CL}\Box$, an homomorphism from the set of terms of $\mathcal{L}\Box$ and the set of elements of $\mathcal{CL}\Box$ need to be defined.

The homomorphism from $\mathcal{L}\Box$ into $\mathcal{CL}\Box$ is sketched below. It allows us to associate to each term T of $\mathcal{L}\Box$ its *structural normal form* denoted *snf*(T) . This homoporphism takes into account axioms of $EQ_\Box$ and the normalization strategy choosen. Indeed, to obtain a normal form many normalization strategies may be applied (*e.g.* deletion of redundant information). We chose to add implicit information in the classical part, this strategy is a kind of partial saturation which is a trick largely used to make easier subsumption and lgg computation. On the other hand, for complexity reasons we only keep maximal subsets in the $\Box$ part. The $\Box$ part is a Sperner family i.e., an antichain in the inclusion lattice over the power set of $\mathcal{L}_P$ (for more details see [2]).

Homomorphism snf from $\mathcal{L}\Box$ into $\mathcal{CL}\Box$:

Term of $\mathcal{L}\Box$	**Element in $\mathcal{CL}\Box$**
$\top$	$t = <\emptyset,\{\emptyset\}>$
$\bot$	$bo = <P,\{P\}>$
$a \in P$	$<\{a\},\{\emptyset\}>$
$F1 \wedge F2$	$snf(F1) \vee_s snf(F2)$
$\Box F$	$<E_F,\{E_F\}>$

where $snf(F) =< E_F,\emptyset >$ (the $\Box$ part is empty since there is no $\Box$ nesting).

$\vee_s$ is the join operator in $\mathcal{CL}\Box$. It uses the classical union set operator and $\vee_a$ which represents the union operator in antichains.
Let $K1$ and $K2$ be two antichains:

$$K1 \vee_a K2 = \{x \in K1 \cup K2 \mid \nexists y \in K1 \cup K2 \text{ s.t. } x \subset y\}$$

Let $snf(Fi)$ be $< E_{Fi}, Ki >$ for $i = 1, 2$:

$$snf(F1) \vee_s snf(F2) =< E_{F1} \cup E_{F2}, K1 \vee_a K2 >$$

Example 4. $snf((a \wedge b) \wedge \Box(b \wedge c) \wedge \Box b \wedge \Box(c \wedge d)) =< \{a, b, c, d\}, \{\{b, c\}, \{c, d\}\} >$ c and d are added in the classical part according to axiom 6 and 7, $\{b\}$ is removed from the $\Box$ part since $\{b\} \subseteq \{b, c\}$.

Definition 13 (Structural Subsumption in $\mathcal{CL}\Box$). *Let S1 and S2 be two elements of $\mathcal{CL}\Box$,*

S1 $\preceq_s$ S2 (i.e. S1 subsumes S2) iff S1 $\vee_s$ S2 = S2

Proposition 8. *Subsumption in $\mathcal{L}\Box$ is equivalent to structural Subsumption in $\mathcal{CL}\Box$: Let F1, and F2 be two terms of $\mathcal{L}\Box$,*

F1 $\preceq$ F2 iff snf(F1) $\preceq_s$ snf(F2)

5.3 The Galois Lattice $G_\Box$

We consider the two following posets : $(\mathcal{P}(O),\subseteq)$ and $(\mathcal{CL}\Box, \preceq_s)$. In order to obtain a Galois connection between the two posets, we need to define the least general generalisation in $\mathcal{CL}\Box$. The *lgg* uses the following definition of $\wedge_a$ the intersection operator in antichain: any antichain A corresponds to a lower set $L_A = \{x \in A \mid \exists y \in A \text{ s.t. } x \subseteq y\}$. Let A and B be two antichains and L_A and L_B their lower sets:

$$A \wedge_a B = \{x \in L_A \cap L_B \mid \nexists y \in L_A \cap L_B \text{ s.t } .x \subset y\}$$

Definition 14 (Least General Generalisation in $\mathcal{CL}\Box$). *Let S1 = < E1,K1> and S2 = < E2, K2 > be two elements in $\mathcal{CL}\Box$:*

lgg(S1,S2) = <E1 $\cap$ E2,K1 $\wedge_a$ K2>

Let EnsS be a finite set of elements of $\mathcal{CL}\Box$ = {s1,s2,. . . ,sn}:

lgg(EnsS) = lgg(s1,lgg(s2,. . . lgg(sn-1,sn)))

Each instance o is described by a term T of $\mathcal{CL}\Box$. In order to obtain a unique most specific description in $\mathcal{CL}\Box$ satisfied by o, we simply use *snf*(T) denoted $d(o)$ in the following.

Proposition 9. *Let ($\mathcal{CL}\Box$, $\preceq_s$) and ($\mathcal{P}(O)$,$\subseteq$) be two posets. The pair (ext,int) defined by:*

$$\forall\, S \in \mathcal{CL}\Box,\ ext(S) = \{o \in O \mid S \preceq_s d(o)\}$$
$$\forall\, E \in \mathcal{P}(O),\ int(E) = lgg(d(E)) \text{ with } d(E) = \{\, d(o) \mid \forall\, o \in E\}$$

defines a Galois connection between ($\mathcal{CL}\Box$, $\preceq_s$) and ($\mathcal{P}(O)$,$\subseteq$)

Proof : as stated in proposition 4 p11 in [15], (ext,int) is a Galois connection iff:

$$S \preceq_s int(E) \Leftrightarrow E \subseteq ext(S)$$

1) $S \preceq_s int(E) \Rightarrow S \preceq_s lgg(d(E))$
$\forall o \in E, lgg(d(E)) \preceq_s d(o)$ since $lgg(d(E)) = lgg(d(o), lgg(d(E - \{o\}))$
$\Rightarrow S \preceq_s d(o)$ since $S \preceq_s lgg(d(E)) \Rightarrow o \in ext(S)$ by definition of *ext* $\Rightarrow E \subseteq ext(S)$
2) $E \subseteq ext(S) \Rightarrow (\forall o \in E \Rightarrow o \in ext(S)) \Rightarrow \forall o \in E, S \preceq_s d(o) \Rightarrow S \preceq_s int(E)$

Proposition 10. *The extension-intension lattice $G_\Box$ defined by the Galois connection of proposition 9 is denoted as an extended abstract lattice.*

When considering a given extensional abstraction A, we have now a "pure" extensional abstract lattice G_A, as defined in section 4 together with the new intension/extension lattice $G_\Box$ defined on the connection between $\mathcal{CL}\Box$ and $\mathcal{P}(O)$ and relying, for its modalized part, on $ext(\Box t) = p_A \circ ext(t)$. We are now interested in the exact relations between these lattices and the original concept lattice G relating the language $\mathcal{P}(P)$ to $\mathcal{P}(O)$. We first remark that $\mathcal{P}(P)$ is isomorphic to an abstraction of $\mathcal{CL}\Box$ as the classical part of $\mathcal{CL}\Box$ is closed under the lgg operator. Therefore G is obtained as an intensional abstraction of $G_\Box$. A second remark is that the extensional abstract lattice $p_A(G_\Box)$ is isomorphic to G_A: consider some intension t in G, its abstract extension $e = ext_A(t)$ rewrites as $ext(\Box t)$, and the representation of $\Box t$ is the most specific element in $\mathcal{CL}\Box$ whose extension contains e. As a consequence, G_A and G are both less abstract than $G_\Box$. We draw Figure 2 these three lattices in a very simple example with two atomic properties X and Y, 4 instances and the following abstraction $A_I = \{\{1,2\},\{2,3\},\{3,4\}\}$. Note that in the table hereunder, the columns $\Box t$ represent the abstract extensions $ext_A(t)$. For instance the column $\Box XY$ represents the empty set since $ext(XY) = \{2\}$ and that no abstract instance of A_I is included in $ext(XY)$.

Instance	X	Y	□ X	□ Y	□ XY	{1,2}	{2,3}	{3,4}
1	1	0	1	0	0	1	0	0
2	1	1	1	1	0	1	1	0
3	0	1	0	1	0	0	1	1
4	1	0	0	0	0	0	0	1

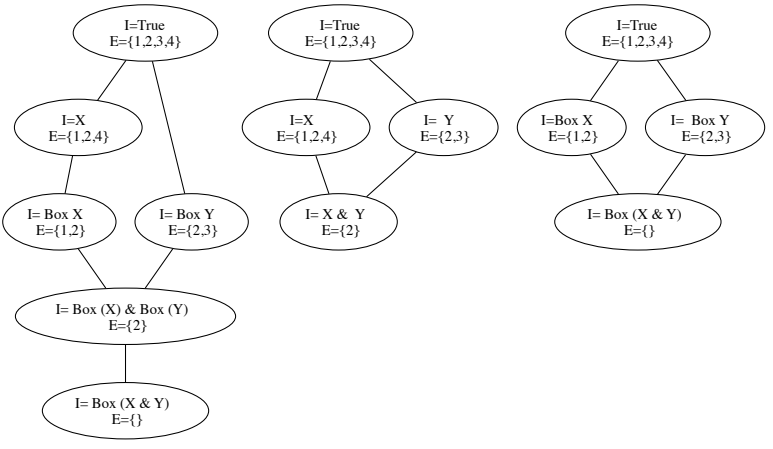


Fig. 2. The three intension/extension lattices corresponding to the example of section 5.2. The node label is a minimal representation of the intension I (here *Box* and *&* stand for $\Box$ and $\wedge$) together with its extension E in $\{1, 2, 3, 4\}$. The leftmost lattice is $G_\Box$, the centermost lattice is the original concept lattice G, and the rightmost lattice is the projected lattice $p_A(G_\Box)$ isomorphic to the abstract lattice G_A.

6 Related Work and Conclusion

In this paper we have proposed abstractions as reductions that preserve the extension-intension lattice structure.They are simply defined as parts of either the intensional language or the extensional space that are closed under the *join* operator. A second contribution is the investigation of extensional abstractions. We have shown that defining an extensional abstraction $A \subseteq \mathcal{P}(O)$ consists in *a priori* defining as units particular subsets of instances, denoted as abstract instances, so applying a change in extensional granularity. A noticeable effect of so preserving the Galois lattice structure is that validity of implications is preserved through extensional abstractions. Finally, we interpret abstract implications as classical implications between two modalized terms. This leads to define *abstract modal logics* and to define an *extended abstract concept lattice* relating $\mathcal{P}(O)$ to a language whose elements have abstract and non abstract parts. We can then search for concepts as "*Oviparous and abstractly apt-to-fly*" where "*o* satifies *abstractly apt-to fly*" means that all the instances of some minimal abstraction of o share this property. Regarding the implementation of abstract concept lattices, we can benefit from implementations of alpha lattices, and in particular their incremental construction [23] adapted from [26], and a software, based on `Galicia` [24], is available[3]. However, *extended abstract concept lattice* construction still has to be investigated, both from an algorithmic and practical point of view. Note that as *extended abstract lattices* belongs to the class of logical concept lattices [13] yet there exists a way to implement them.

[3] `http://www-lipn.univ-paris13.fr/~ champesme/alphabetagalicia`

Acknowledgments. We are grateful to Auguste Forge for useful comments and careful reading of this paper.

References

1. Ben-David, S., Ben-Eliyahu-Zohary, R.: A modal logic for subjective default reasoning. Artif. Intell. 116(1-2), 217–236 (2000)
2. Bollobás, B.: Combinatorics: set systems, hypergraphs, families of vectors, and combinatorial probability. Cambridge University Press, New York (1986)
3. Brézellec, P., Soldano, H.: Tabata: a learning algorithm performing a bidirectional search in a reduced search space using a tabu stratégie. In: Eur. Conf. in Art. Int., ECAI 1998, Brighton, England, pp. 420–424. Wiley and Sons, Chichester (1998)
4. Caspard, N., Monjardet, B.: The lattices of closure systems, closure operators, and implicational systems on a finite set: a survey. Discrete Appl. Math. 127(2), 241–269 (2003)
5. Cattaneo, G., Ciucci, D.: Lattices with interior and closure operators and abstract approximation spaces. T. Rough Sets 10, 67–116 (2009)
6. Chaudron, L., Maille, N.: Generalized formal concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867, pp. 357–370. Springer, Heidelberg (2000)
7. Chellas, B.F.: Modal Logic: an introduction. Cambridge University Press, Cambridge (1980)
8. Davvaz, B., Mahdavipour, M.: Rough approximations in a general approximation space and their fundamental properties. International Journal of General Systems 37(3), 373–386 (2008)
9. Diday, E., Emilion, R.: Maximal and stochastic galois lattices. Discrete Appl. Math. 127(2), 271–284 (2003)
10. Dionne, R., Mays, E., Oles, F.J.: The equivalence of model-theoretic and structural subsumption in description logics. In: Int. Joint Conf. on Art. Int. (IJCAI), pp. 710–717 (1993)
11. Erné, M., Koslowski, J., Melton, A., Strecker, G.E.: A Primer on Galois Connections. Annals of the New York Academy of Sciences 704(Papers on General Topology and Applications), 103–125 (1993)
12. Ferré, S.: Negation, opposition, and possibility in logical concept analysis. In: Missaoui, R., Schmidt, J. (eds.) ICFCA 2006. LNCS (LNAI), vol. 3874, pp. 130–145. Springer, Heidelberg (2006)
13. Ferré, S., Ridoux, O.: An introduction to logical information systems. Information Processing and Management 40(3), 383–419 (2004)
14. Ganascia, J.-G.: TDIS: an algebraic formalization. In: Int. Joint Conf. on Art. Int (IJCAI), vol. 2, pp. 1008–1013 (1993)
15. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
16. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 129–142. Springer, Heidelberg (2001)
17. Guigues, J.L., Duquenne, V.: Famille non redondante d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines 95, 5–18 (1986)
18. Liquière, M., Sallantin, J.: Structural machine learning with Galois Lattice and Graphs. In: ECML 1998, pp. 305–313. Morgan Kaufmann Publishers Inc., San Francisco (1998)
19. Pasquier, N., Taouil, R., Bastide, Y., Stumme, G., Lakhal, L.: Generating a condensed representation for association rules. Journal of Intelligent Information Systems (JIIS) 24(1), 29–60 (2005)

20. Pernelle, N., Rousset, M.-C., Soldano, H., Ventos, V.: Zoom: a nested Galois lattices-based system for conceptual clustering. J. of Experimental and Theoretical Artificial Intelligence 2/3(14), 157–187 (2002)
21. Saitta, L., Zucker, J.-D.: A model of abstraction in visual perception. Applied Artificial Intelligence 15(8), 761–776 (2001)
22. Soldano, H.: A modal view on abstract learning and reasoning. Technical report, L.I.P.N, UMR-CNRS 7030, Université Paris-Nord (2010)
23. Soldano, H., Ventos, V., Champesme, M., Forge, D.: Incremental construction of alpha lattices and association rules. In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010. LNCS, vol. 6277, pp. 351–360. Springer, Heidelberg (2010)
24. Valtchev, P., Grosser, D., Roume, C., Hacene, M.R.: Galicia: An open platform for lattices. In: In Using Conceptual Structures: Contributions to the 11th Intl. Conference on Conceptual Structures (ICCS 2003), pp. 241–254. Shaker Verlag (2003)
25. Valtchev, P., Missaoui, R., Godin, R.: Formal concept analysis for knowledge discovery and data mining: The new challenges. In: Eklund, P.W. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 352–371. Springer, Heidelberg (2004)
26. Valtchev, P., Missaoui, R., Godin, R., Meridji, M.: Generating frequent itemsets incrementally: two novel approaches based on Galois lattice theory. J. of Experimental and Theoretical Artificial Intelligence 2/3(14), 115–142 (2002)
27. Ventos, V.: A deductive study of the c-classicde description logic. In: Description Logics. AAAI Technical Report, vol. WS-96-05, pp. 192–196. AAAI Press, Menlo Park (1996)
28. Ventos, V., Soldano, H.: Alpha Galois Lattices: An Overview. In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 299–314. Springer, Heidelberg (2005)
29. Yao, Y.Y., Lin, T.Y.: Generalization of rough sets using modal logics. Intelligent Automation and Soft Computing 2, 103–120 (1996)

Exploring the Information Space of Cultural Collections Using Formal Concept Analysis

Tim Wray and Peter Eklund

School of Information Systems and Technology
University of Wollongong
Northfields Ave, Wollongong, NSW 2522, Australia
twray,peklund@uow.edu.au

Abstract. Within the cultural informatics community, there is a strong desire to mine and understand relationships within and among collections of objects. In this paper we describe a case study of applied Formal Concept Analysis to cultural heritage and art collections. We base our inter-disciplinary research on our development of a navigation framework that drives the Virtual Museum of the Pacific – an FCA-based application that employs a conceptual neighbourhood paradigm for browsing concept lattices. We also utilise a feature called conceptual similarity that allows users to search for similar objects and hence promote knowledge discovery of the objects within the collection. We describe how we can construct a meaningful information space derived from museum documentation while considering complexity and associated performance issues of large formal contexts. We report the resulting lattice structure, user experience and relevance of our FCA-based application in browsing and exploring objects from a cultural domain. Our research is an applied case study of term extraction and context creation based on data-sets from the Australian Museum and Powerhouse Museum collections.

1 Introduction

Within digital collections, the ability to cluster and find relationships between objects is seen as a desirable property. An object's meaning can be significantly enhanced when it is supplemented with contextual cues and otherwise hidden relationships, particularly as collections grow in size and diversity. When applied to art and cultural collections, and specifically towards collections that are geared towards a public or stakeholder audience, an alternate interaction paradigm that focuses on serendipitous rather than direct search can represent a dynamic view of collections rather than a fixed hierarchy or a 'locked down search box.' Formal Concept Analysis, with its ability to generate conceptual hierarchies inherent within data, is a promising approach towards unlocking the value of art and cultural collections. Novel approaches towards presentation fuel stakeholder interest and curiousity, and in effect, can ultimately motivate the digitisation of the endless array of cultural content that is otherwise locked up and not on public display.

P. Valtchev and R. Jäschke (Eds.): ICFCA 2011, LNAI 6628, pp. 251–266, 2011.

This paper represents a snapshot in an on-going research project to develop a framework that allows a unique and explorative approach towards browsing cultural collections. We focus our problem domain towards art and cultural collections, and highlight the specific technicalities of term extraction and context creation, and describe our conceptual browsing, search and similarity features in Section 3 as they apply to these extracted terms. This paper focuses attention on the need to automate the formal context creation process for large collections of digital content. More importantly is the need to create such formal contexts with a low margin of error or attribute ambiguity as it is well known that a noisy data-set can result in lattice structures that are prohibitively unreadable, complex and disjointed.

When considering the nature of museum documentation and cultural heritage sources, building suitable formal contexts for lattice generation can present some particular challenges. These data-sets suffer from particular problems – these include: inconsistent or incorrect use of terminology (e.g. 'domicile' instead of 'domestic'); multiple synonyms that represent the same term; different uses of specialised terminology across institutions and even individual cataloguers and a general lack of maintenance of 'exhibition quality' meta-data due to the vast size of collections, legacy systems and the budgetary constraints that museums operate under. Large scale cultural heritage portals such as Europeana[1] and Digital NZ[2] compound the problem by aggregating meta-data from multiple institutions, each with their own fields, standards, disciplines and approach towards documentation. As museums and cultural heritage portals trend towards the aggregation and combination of user-generated meta-data, these factors pose particular challenges towards constructing normalised, stable and relatively clean formal contexts from their collections.

The paper is structured as follows: first, in Section 2, we will present a brief overview of Formal Concept Analysis and some of the its underlying techniques for data analysis and discovery. In Section 3, we describe the key features of the Virtual Museum of the Pacific project, an instance of an evolving software framework for collection browsing and discovery. These design features were engineered to allow conceptual browsing, discovery, search and similarity within cultural collections. We then focus the discussion towards our approach and results for a term extraction method in Section 4 as they are applied to the our software framework.

2 Formal Concept Analysis

Formal Concept Analysis is a data analysis technique that allows the synthesis of formal concepts based on a collection of objects and their attributes. It follows the philosophical tradition that any concept or unit of thought could be understood in terms of its attributes (or intension) and its objects that are

[1] http://www.europeana.eu/portal/
[2] http://www.digitalnz.org/

characterised by those attributes (its extension). A more expansive overview of Formal Concept Analysis is provided in [1].

Formal concepts can be placed into a specialisation hierarchy, where more specific concepts (with fewer objects and more attributes) can be viewed as a specialisation of other less specific formal concepts. The result is an algebraic structure known as a concept lattice. We exploit the properties of the concept lattice – along with techniques such as conceptual scaling – to drive the navigation and design features of our software framework.

2.1 Conceptual Scaling

Conceptual scaling is a technique that allows stored views of the data being analysed [2]. Conceptual scales encompass specific attribute sets and are represented by a mathematical entity called a formal context. A formal context is a triple (G, M, I) where G is a set of objects, M is a set of attributes and I is a relation between the objects and the attributes, called an incidence relation. Within our applications, we use conceptual scales to represent specific contexts with different themes such as 'works as described by materials' or 'works as described by origin.' In essence, conceptual scales can be used to represent sub-contexts of an entire collection, both in order to reduce navigational and computational complexity. To an end-user, they can be used to create multiple views or 'lenses' on the information space. We describe our implementation of conceptual scaling in Section 3.1.

2.2 The Conceptual Neighbourhood Paradigm

We apply the conceptual neighbourhood paradigm for browsing the information space provided by the concept lattice – which in turn is derived by conceptual

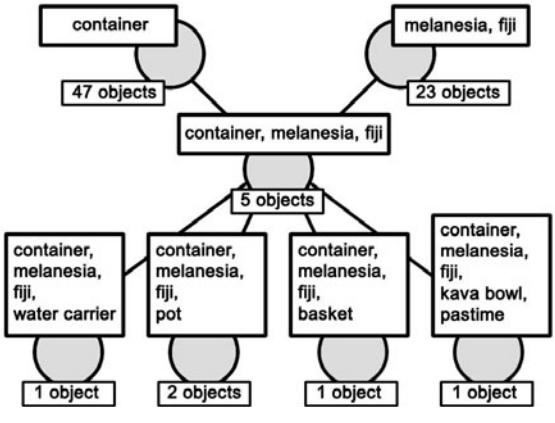


Fig. 1. The conceptual neighbourhood representation of a formal concept of attributes { 'container', 'melanesia', 'fiji' } – shown here are its upper neighbours (top row) and lower neighbours (bottom row)

scaling from the given data. Within this approach, the user is placed at a single formal concept within the lattice. Users can move from one formal concept to another by incrementally navigating across upper and lower neighbours. Fig. 1 shows a lattice neighbourhood representation. Evident are the upper and lower neighbours for the formal concept where its attributes are represented by { 'container', 'melanesia', 'fiji' } – users can effectively generalise or specialise their view on the collection by navigating to neighbouring concepts. We describe our implementation of this approach in Section 3.1.

2.3 Conceptual Similarity

We exploit a feature called conceptual similarity that allows us to find neighbouring concepts for a given formal concepts, or related concepts where a certain set of attributes do not manifest in a single object. Our approach uses variations on defined distance and similarity metrics in the FCA literature [3] in order to find relevant concepts.

The similarity metric we applied uses the number of common objects and the number of common attributes of two given formal concepts (A, B) and (C, D):

$$similarity((A, B), (C, D)) := \frac{1}{2}\left(\frac{|A \cap C|}{|A \cup C|} + \frac{|B \cap D|}{|B \cup D|}\right).$$

The distance metric uses the size of the total overlap of the intent and extent normalised against the total size of the context. For two concepts (A, B) and (C, D):

$$distance((A, B), (C, D)) := \frac{1}{2}\left(\frac{|A \setminus C| + |C \setminus A|}{|G|} + \frac{|B \setminus D| + |D \setminus B|}{|M|}\right).$$

A predecessor program to the Virtual Museum of the Pacific – ImageSleuth2 – applies distance and similarity metrics to traverse the concept lattice and present ranked formal concepts with similarity metrics for a given object and its set of attributes. This allows for a query by example feature which we incorporate into the Virtual Museum of the Pacific – details and examples on the nature of lattice traversal and the method that formal concepts are selected and ranked can be found in [4].

3 The Virtual Museum of the Pacific

The Virtual Museum of the Pacific, an FCA-based application previously reported in [5], is a collaborative project between the University of Wollongong and the Australian Museum that leverages a collection of 427 objects from the Australian Museum's Pacific collection.[3] From a design perspective, it is arguably the successor of the Sleuth series reported in [6], [4] and [7]. Our work

[3] The Virtual Museum of the Pacific is available in two versions : http://epoc.cs.uow.edu.au/vmp is the version that is based on the manually selected set of attributes. http://epoc3.cs.uow.edu.au/vmp is an alternate version based on an automatically generated set of attributes.

extends the FCA-based browsing features developed in the Sleuth series by incorporating collaborative tagging, the ability to add rich media, an overhaul of its user interface (reported in [8]), an and implementation of query expansion, content-based retrieval and term extraction described in Sections 3.2, 3.3 and 4 respectively. As a continuously evolving project that represents the current feature-set of our framework for term extraction, search and navigation, we will briefly describe its features in this section, with particular reference to its FCA-based browsing features. Section 4 elaborates on its design in more detail by highlighting issues relating to its term extraction and context creation features.

3.1 Navigation Using Formal Concept Analysis

Navigation within the Virtual Museum of the Pacific uses the conceptual neighbourhood approach for browsing collections as described in Section 2.2. Using this approach, objects of a single formal concept are represented as thumbnails and users can navigate the conceptual neighbourhood by incrementally moving to upper or lower neighbours within the user interface, shown respectively at the top and bottom of the screen in Fig. 2. This allows users to easily generalise or refine their conceptual view on the collection. For instance, in Fig. 2, a user can refine their conceptual neighbourhood view to present all Melanesian wooden body accessories by clicking on the 'melanesia' link (located bottom-centre), or by viewing only wooden objects by clicking on the 'body accessories' link (located top-centre) and removing it from the search space.

In Fig. 2, the left-hand side of the screen shows a list of conceptual scales which are called 'perspectives' within the Virtual Museum of the Pacific. Perspectives either represent logical groupings of objects (e.g. 'containers', 'hunting fishing weapon' and 'transport' objects) or facets that circumscribe a specific dimensions (e.g. 'origin' and 'materials'). Perspectives reduce the navigational and computational complexity by circumscribing attributes (and their corresponding objects) into thematically defined formal contexts. To an end-user, a perspective represents a specific 'lens' on a collection, and perspectives can be combined to effectively query the data, allowing for multi-faceted browsing of the collection.

3.2 Text Search Using Query Expansion

For the purpose of creating useful concept lattices for the objects of the Virtual Museum of the Pacific, it is desirable to employ a set of unambiguous terms as the set of formal attributes. Within the Virtual Museum of the Pacific, a set of 609 terms (from the Australian Museums own control vocabulary of 1198 terms) is created to represent the attributes of the 427 Pacific objects. These terms are narrow, specific and unambiguous, and while they allow the construction of stable lattices, the use of these attributes alone for keyword-based search yields poor recall due to the concise and often highly specific terminology employed by controlled vocabularies. In an effort to align the Virtual Museum of the Pacific (and its associated software framework) as an easily accessible repository of

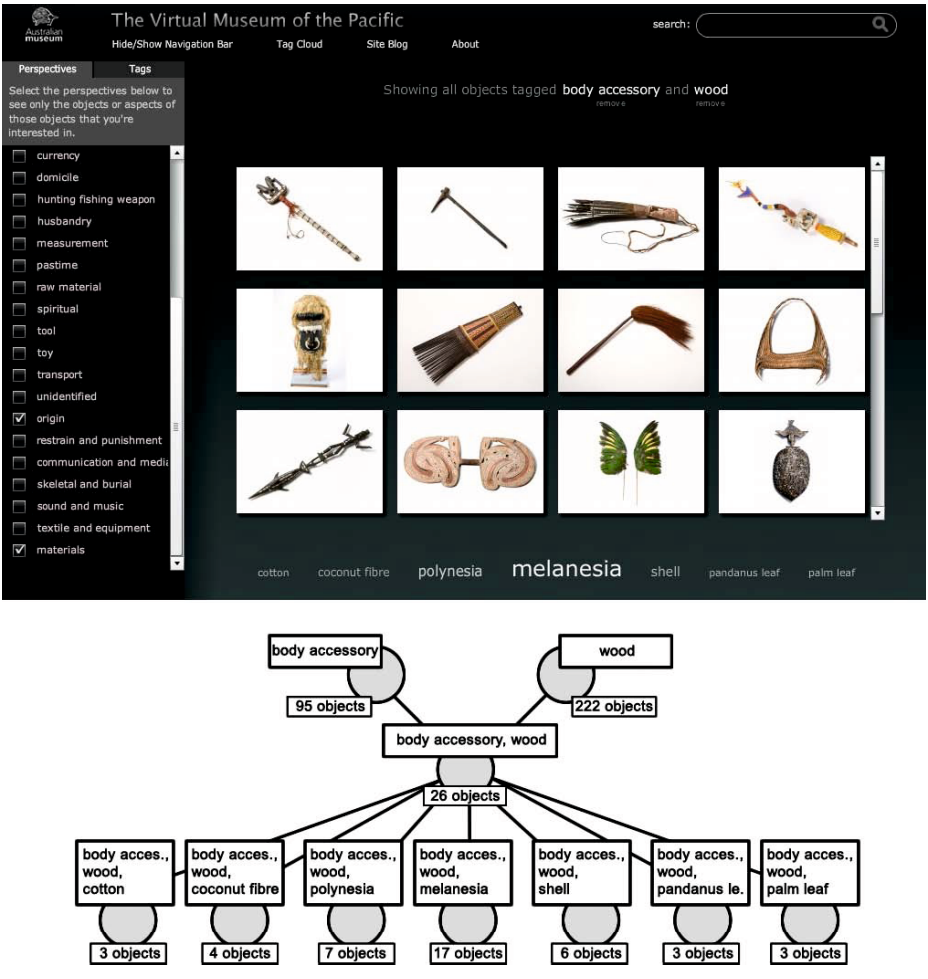


Fig. 2. The main navigation interface of the Virtual Museum (top) of the Pacific, showing the conceptual neighbourhood view of all objects that have 'body accessory' and 'wood' attributes (bottom)

cultural objects, we have implemented WordNet-based [9] query expansion to facilitate free-text searching for formal concepts.[4]

In our implementation of WordNet-based query expansion, the search string "Melanesian objects made of bone" returns the following three formal concepts:

- 6 objects tagged bone and melanesia
- 6 objects tagged tooth and melanesia
- 1 object tagged pigment, melanesia, natural fibre, bougainville, bougainville island, teeth

[4] This feature is available within the Virtual Museum of the Pacific, however, you can access a lightweight HTML version at http://epoc.cs.uow.edu.au/vmpsearch/

We perform query expansion by creating a set of extended terms for a given attribute from the formal context (we call this the *base term*). These extended terms are used solely for the purposes of expanding user queries and are not considered part of the formal context. To obtain this set of extended terms, the correct word-sense – a distinct and logical grouping of synonyms – is selected from WordNet. We follow a relatively simple approach for word-sense disambiguation where we select the word-sense whose terms have the highest occurrence of matches within the Virtual Museum of the Pacific's corpus of detailed object text labels, and then use those terms as a set of extended terms for a given base term.

When a user query is processed, the algorithm removes all stop-words and any other terms from a pre-defined list that do not represent specific objects or qualities (such as 'object', 'artifact' etc.). We then stem these terms and map them against our extended term set, retrieving a set of base terms for the given query. For instance, the search phrase "Fijian house items" (shown in Fig. 3) returns base terms { 'fiji', 'house model' and 'domicile object' }. The algorithm then attempts to find a matching formal concept for these given base terms, and if none is found, the most similar formal concept is retrieved, using the conceptual similarity metrics described in Section 2.3. For example, for a given query "Fijian objects that are made with metal and stone", no exact matches are found as there are no such objects within the collection. Instead, the algorithm returns two similar yet disjointed concepts: "all Fijian objects made with stone, coconut fibre

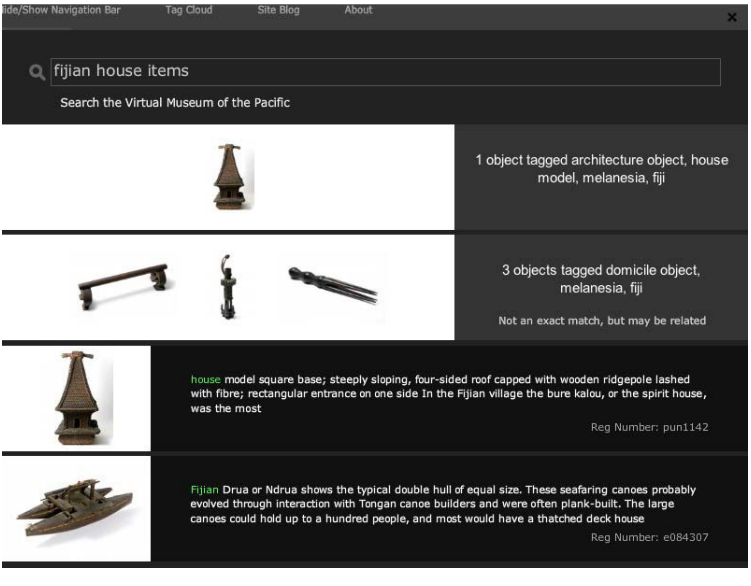


Fig. 3. The Virtual Museum of the Pacific allows for free text searching that can return both matching formal concepts (or their approximations) and their individual objects. Here we see the matching formal concept for the search phrase "Fijian house items"

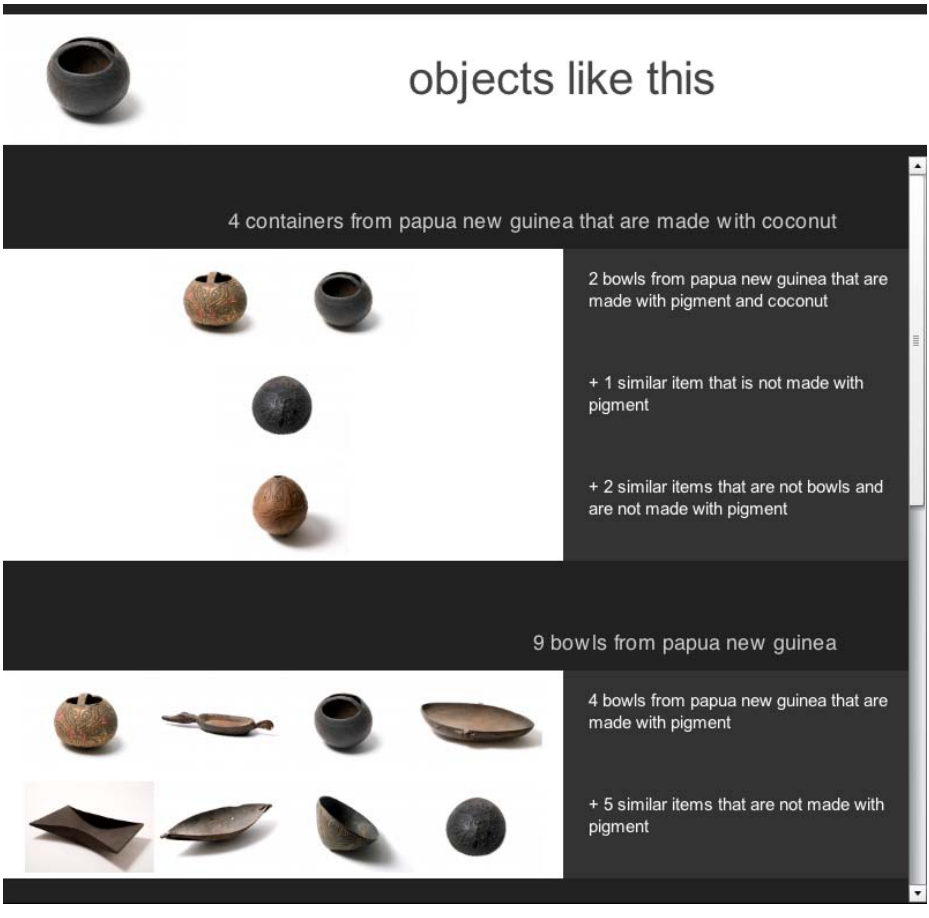


Fig. 4. Demonstration of content-based retrieval within the Virtual Museum of the Pacific. Similar concepts are clustered, then expressed in natural language.

and wood" and "all objects made with metal." To minimise the computational overhead of the search algorithm, a formal sub-context based on the minimal union of conceptual scales is created before it checks it for matching concepts, or searches for similar concepts. Following the running "Fijian house items" example, the extracted base terms { 'fiji', 'house model' and 'domicile object' } match the 'origin', 'architecture' and 'domicile objects' scales respectively. The reduced sub-context is then used to perform concept matching and conceptual similarity operations, rather than use the entire full context of 427 objects and 605 attributes.

3.3 Content-Based Retrieval Using Conceptual Similarity

The Virtual Museum of the Pacific also allows users to view a single object, and find related objects that share similar properties or qualities to that object. In

addition, it can also describe how those objects are related in natural language. We employ techniques of conceptual similarity and natural language generation to achieve this.

For a given object, we derive similar concepts based on the object's attributes using conceptual similarity methods described in Section 2.3. In essence, we gather formal concepts from the conceptual neighbourhood whose attributes most closely match the object we are comparing. Due to the high intersection of objects from formal concepts that are 'close' to one another within the conceptual neighbourhood, we then cluster formal concepts that have a set of common attributes. For instance, as shown in Fig. 4, the first 'result' actually links to three formal concepts, each one within the list being a super-concept of the one before it. These results are then expressed in natural language where an attribute's membership with one or more conceptual scales implies the relationship type that attribute has with the object. For example, if an object has attributes 'wood' and 'steel', and those attributes are within the 'materials' conceptual scale, then it can be given that the object is made of wood and steel. Likewise, it could be said that an object is from Papua New Guinea if the attribute 'papua new guinea' is a member of the 'origin' conceptual scale. Although this form of semantic expression lacks a certain explicitness compared to other formal descriptive languages, we theorise that it is particularly well suited to the cultural heritage domain as its terms and facets often imply a specific relationship-type to the object at hand, and as such it also benefits from its simplicity in that it doesn't require the use of complex descriptive languages to store its associations. We test this assumption as we perform term extraction on a different domain within Section 4.

4 Building Formal Contexts Using Term Extraction

The idea of using automated term extraction to annotate and group objects has been discussed extensively. Much of these discussions centre around the use of lexical resources to help normalise and categorise extracted terms, or to automatically create faceted hierarchies based on a large corpus of free text. For instance, the approach by Stoica and Hearst [10] uses WordNet [9] to offer a hierarchical view of topics covered within a videoconference discussion, much in the same way that we construct hierarchies using the Getty's Art and Architecture vocabularies. Previous studies have shown that using an external thesuarus's hypernym structure is a good way of building faceted hierarchies [11], [12]. It was shown that WordNet hypernyms and synonyms have high precision, and therefore are useful in accurately (and unambiguously) describing objects, but are poorly suited to be used as index terms for search due to their low recall.

Closely related to our approach for term extraction is the work by Klavans et al. [13]. They describe a system that provides a machine-assisted way of image cataloguers to assign subject terms to image collections, where these terms are extracted from high-quality image descriptions and related texts. Their approach for semi-automated term extraction uses part-of-speech tagging, extraction and

vocabulary alignment techniques against the Getty Art and Architecture thesaurus to help cataloguers tag terms to visual resource collections. Interestingly, they also discuss some of the challenges and problems associated with word-sense disambiguation, and ensuring that the semantics and meaning from free textual data are well understood and interpreted so the appropriate terms are assigned. These challenges also apply to our approach to term extraction, which in turn, may affect the quality of browsing, search and content-based retrieval.

Our design goal for a term extraction process is to make it as autonomous as possible so that formal contexts and conceptual scales can be created from a large corpus of art objects and their meta-data. To evaluate our approach, we apply our method to two data-sets. The first data-set is based on the object descriptions and labels from the 427 objects from the Virtual Museum of the Pacific. Within this data-set, the object descriptions are thorough, well researched and documented and are written by a single anthropologist. Thematically speaking, the object descriptions are generally well focused on the provenance of the object and its use within a cultural context with a very large emphasis on its physical materials and properties. The collection represents a focused, narrow view of a specific type of object (Pacific cultural artefacts, although the objects themselves represent a very diverse range of Pacific cultures, materials and object types) but most importantly the source meta-data for our context creation is consistent, with similar usage of terms and language across all object labels and descriptions.

To assess the generality of our approach, we also apply our method to 427 objects from the Powerhouse Museum. These objects, available via an API[5], come from a diverse range of categories such as agricultural equipment, books, computers, photographs and costumes. Some of the object descriptions are recent, while others date back to over 100 years. To represent a diverse collection of objects, records and meta-data, five to six objects were selected from each of its 75 categories.

Our experimentation effectively compares three formal contexts: the carefully selected, manually generated set of terms from the Virtual Museum of the Pacific[6] reported in [14]; a generated set of terms based on high quality meta-data from that same collection[7] and a generated set of terms from a categorically and historically diverse collection.[8] We then compare our conceptual based browsing, search and content-based retrieval features on these three formal contexts.

4.1 The Term Extraction and Mapping Process

We initially extract key terms by parsing object meta-data and descriptions through Yahoo's Term Extraction service. This service retrieves key concepts and terms that are considered to be descriptive of the text at hand, and in

[5] http://api.powerhousemuseum.com/
[6] Available at: http://epoc.cs.uow.edu.au/vmp
[7] Available at: http://epoc3.cs.uow.edu.au/vmp
[8] Available at: http://epoc3.cs.uow.edu.au/phe

terms of its performance, it is considered to be state-of-the-art when compared to baseline tf-idf measures [15].

In isolation, the extracted terms provide a good representation of their objects. However, the use of these terms proved to be problematic when we apply our conceptual-based browsing features. In our initial experimentation, we simply parsed all the objects and their extracted terms and created a new conceptual scale called 'automatically extracted terms.' Performance of navigating these contexts using conceptual-based navigation was degraded severely due to their size and complexity, and even more so as we apply our search and content-based retrieval features. This was due to the very large contexts resulting from the noise induced by the term extraction process – it resulted in a set of 2294 attributes for the Australian Museum's Pacific collection and a set of 3613 attributes for the Powerhouse collection. However, most of these attributes (61.7% and 85.2% respectively) were only tagged to a single object, therefore producing a lattice with many small, single object concepts.

To reduce the complexity of these formal contexts, we have adopted an approach where attributes that have less than N objects assigned to them are removed. From N one can calculate a total percentage, $N/|G|$, and we have set a baseline percentage of 0.25%, so that $N = 2$ (for a set of 427 objects). In effect, we are uncommonly used terms from each of these contexts. More sophisticated complexity reduction techniques, such as removing formal concepts below a stability threshold [16] rely on prior computation of the entire lattice, and hence cannot be applied in our approach due to the size of the formal contexts and the fact that we do not pre-compute the entire lattice for browsing and search.

To further reduce the complexity of the contexts, we map and normalise the terms to the Getty's Art and Architecture Thesaurus. The Art and Architecture Thesaurus (AAT)[9] is an extensive control vocabulary developed by the Getty Research Institute. It contains approximately 34,000 terms that are used to describe and catalogue objects, and these terms are organised into facet hierarchies. There are numerous benefits in doing this: first, all synonymous (and relevant) terms are normalised against a thesaurus. Due to the hierarchical representation of the terms within the AAT, this allows for the construction of attribute hierarchies that can be represented quite naturally within FCA, and as such prove especially useful for operations that rely on conceptual similarity. Secondly, these hierarchies (grouped into 7 facets) can be used to create conceptual scales. This allows for the creation of 'perspectives' described in Section 3.1, and also allows us to apply lightweight semantics to the tags as described in Section 3.3. Finally, and perhaps most importantly, the use of a standardised vocabulary (and a collection-independent term extraction process) allows the significant possibility of federating multiple and distinct collections into an FCA-based browsing applications. This possibility will be explored in future research.

Fig. 5 outlines the process of term extraction and alignment. Automated term-mapping and disambiguation to the AAT, as described in [13], is a particularly challenging task, and the authors describe some techniques for overcoming

[9] http://www.getty.edu/research/tools/vocabularies/aat/about.html

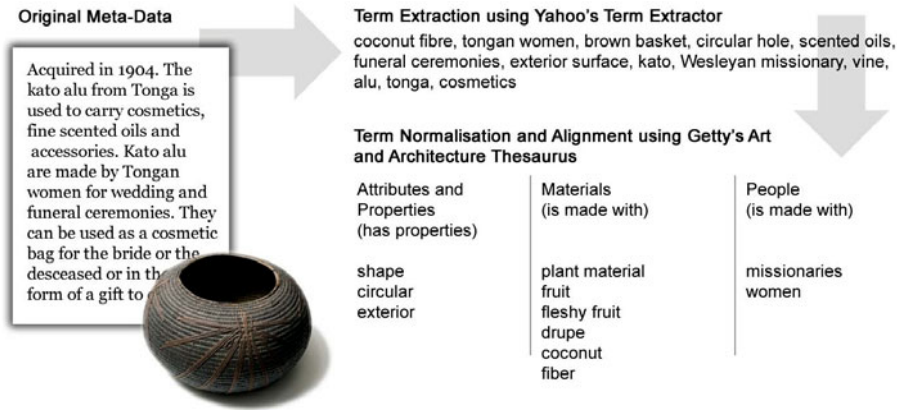


Fig. 5. Attributes are derived from the meta-data records of the object (left) by exxtracting terms using Yahoo's Term Extraction Web Service (top) and then aligning them to the Getty Art and Architecture vocabularies (bottom)

disambiguation problems. For our experiments, we map the terms manually and identify some of the common problems encountered with term mapping process. In some cases, extracted terms such as 'Cook' actually referred to the name 'Captain Cook' rather than the common term 'cook (occupation).' In other cases, the extracted terms provided overly generalised or ambiguous description of an object that provided insufficient semantic or disambiguation information (such as the terms 'artefact', 'standard deviation', 'registration number') or that the terms did not simply exist in the AAT in the correct word-sense that was relevant to the collection. These manually mapped terms will be used as baseline data for future research where we investigate automated alternatives for mapping extracted terms to the AAT.

4.2 Results of Term Extraction

For the Australian Museum's Pacific collection, conceptual navigation of the automatically extracted terms allowed the ability to find object clusters that were non-existent in the formal context of manually extracted terms. For instance, the generation of the *color*, *people*, *design elements* and *visual works* conceptual scales – derived from the AAT – allowed the ability to explore previously obscured facets of objects that were otherwise 'hidden' in museum documentation. The hierarchical nature also allowed the ability to 'drill down' into specific terms and sub-concepts (e.g. users could drill down from metal-based objects to copper-based objects). The Powerhouse collections offered a similar level of user experience, although there were some cases where terms were incorrectly assigned due to the fact that the term refers to a component or feature of the object, rather than the object itself. For instance, the term 'space' was both correctly assigned to spacecraft paraphernalia, but incorrectly assigned to computer keyboard (where, 'space' referred to the 'space-bar' on the computer's

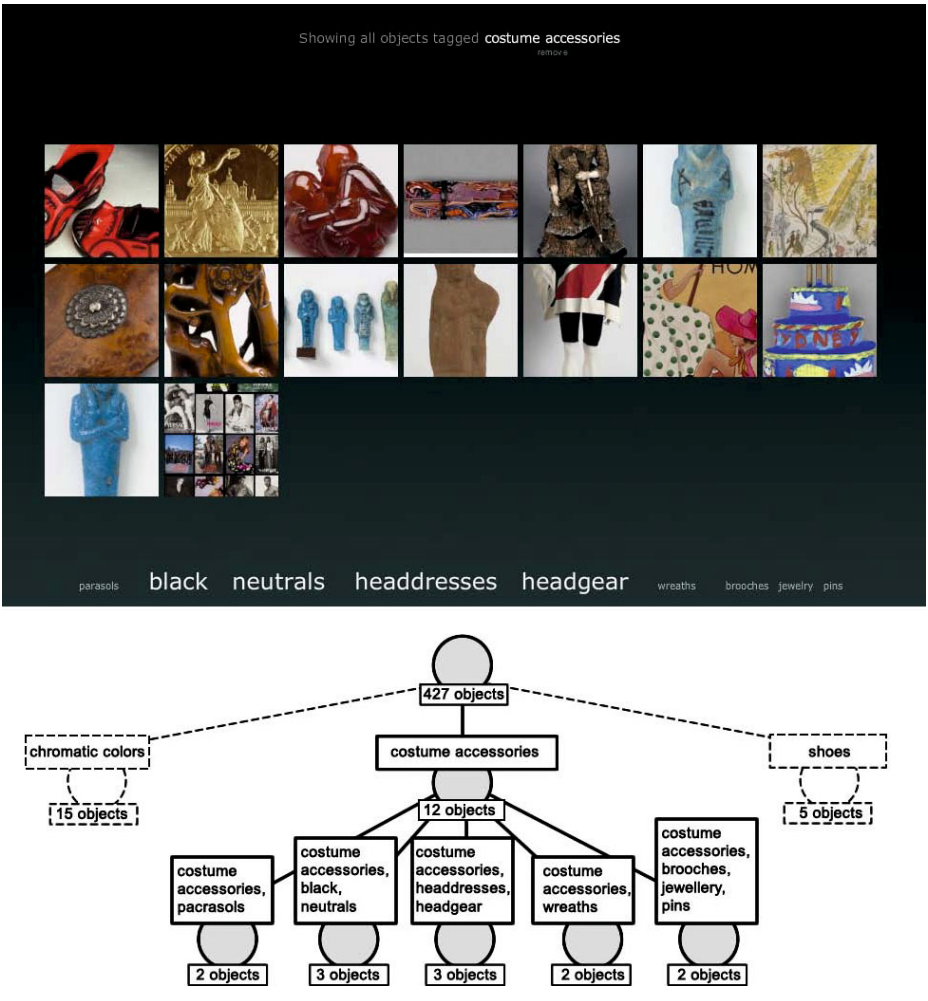


Fig. 6. A subset of the Powerhouse collection (top), rendered as a conceptual neighbourhood (bottom). Here, we can infer that the majority of the costume accessories are black or are headdresses. The formal context for this collection was constructed entirely from the term extraction techniques described in this paper.

keyboard). In some cases, terms were ambiguous or could hold multiple interpretations, where, for instance, 'leaves' referred to both leaf material and leaf designs and motifs. Aside from these isolated cases, the objects within each concept were overall representative of their intension, as shown in Fig. 6.

Content-based retrieval for the extracted term sets generally retrieved objects that were related mostly by their physical composition or use, rather than the actual item type or category of object itself. This feature may allow users to find related objects beyond the well-defined categories that museum collections often impose. Fig. 7 lists the groups of objects that are related to the

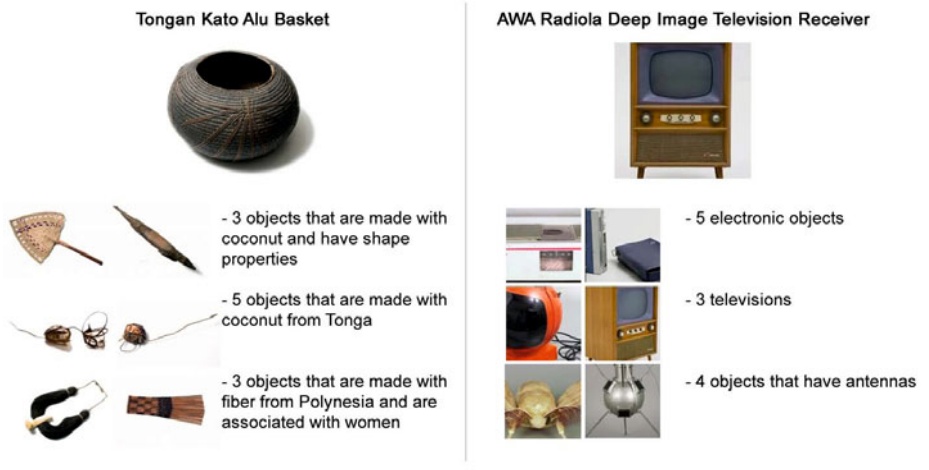


Fig. 7. The results of the content-based retrieval for a kato alu basket from the Australian Museum's Pacific collection and a black and white television receiver from the Powerhouse collection, based on the formal contexts created by the term extraction features as described within this paper

Australian Museum's kato alu basket and the Powerhouse Museum's black and white television. When we compare the content-based retrieval results between the manually-added and automatically extracted data-sets of the Australian Museum's Pacific collection, the results appeared to be less indicative of their object types (e.g. "2 bowls from melanesia that are made with plant fibre") and are usually presented in more abstract terms, yet they are still related according to their materials and function (as shown in Fig. 7). However, the free text search capabilities described in Section 3.2 for the automatically extracted contexts generally returned fewer or more irrelevant formal concepts, particularly for search queries that describe what an object is, rather than its characteristics. For instance, while queries such as "objects made of metal" generally returned related concepts, queries that describe the objects' common nouns, such as "bowls", "containers" or "shoes" generally retrieved results with lower recall or no results at all. This is consistent with the findings of previous research that suggests that term extraction from lexical resources produces terms that have high precision and low recall [11] and that when users search, they tend to user terms that are more broader and less specific (e.g. 'metal' instead of 'copper', or 'clocks' instead of 'chronometers') [17].

5 Conclusion and Future Work

In this paper, we have presented some of the main design features of our software framework for browsing collection content and evaluated an approach for term extraction and context creation using term extraction Web Services and external thesauri. For a collection of 427 objects, navigation of concept lattices using

a set of extracted terms did not present any significant performance or usability problems. Despite some small errors relating to word-sense disambiguation along with the semantics of the extracted terms as they apply to objects, our method for term extraction generally worked well for scale creation and content-based retrieval queries. Based on our generated contexts, our applications showed strength in the ability to browse associatively and relate objects to one another due to the hierarchical nature of the mapped terms and their precision. However, search performance was limited due to low recall of the object names and categories within the museum meta-data records. So far, the results look promising and future research will focus on term extraction and context creation for larger collections and whether such extraction techniques are sustainable for other sources of museum documentation such as secondary documents, articles and news sources and user commentary and stories.

References

1. Wille, R., Ganter, B.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
2. Ganter, B., Wille, R.: Conceptual scaling. In: Roberts, F. (ed.) Applications of Combinatorics and Graph Theory to the Biological and Social Sciences, pp. 139–167. Springer, Heidelberg (1989)
3. Saquer, J., Deogun, J.S.: Concept aproximations based on rough sets and similarity measures. Int. J. Appl. Math. Comput. Sci. 11, 655–674 (2001)
4. Eklund, P., Ducrou, J., Wilson, T.: An intelligent user interface for browsing and searching MPEG-7 images using concept lattices. In: Yahia, S.B., Nguifo, E.M., Belohlavek, R. (eds.) CLA 2006. LNCS (LNAI), vol. 4923, pp. 1–22. Springer, Heidelberg (2008)
5. Eklund, P., Wray, T., Goodall, P., Bunt, B., Lawson, A., Christidis, L., Daniels, V., van Olffen, M.: Designing the Digital Ecosystem of the Virtual Museum of the Pacific. In: 3rd IEEE International Conference on Digital Ecosystems and Technologies, pp. 805–811. IEEE Press, Los Alamitos (2009)
6. Ducrou, J., Eklund, P.: An intelligent user interface for browsing and search MPEG-7 images using concept lattices. Int. Journal of Foundations of Computer Science 19(2), 359–381 (2008)
7. Ducrou, J.: DVDSleuth: A case study in applied formal concept analysis for navigating web catalogs. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 496–500. Springer, Heidelberg (2007)
8. Wray, T., Eklund, P.: Social tagging for digital libraries using formal concept analysis. In: Kryszkiewicz, M., Obiedkov, S. (eds.) Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, pp. 139–150 (October 2010)
9. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
10. Stoica, E., Hearst, M.A.: Nearly-automated metadata hierarchy creation. In: Proceedings of the North American Chapter of the Association of Computational Linguistics, pp. 117–120 (2004)
11. Dakka, W., Ipeirotis, P.G., Wood, K.R.: Automatic construction of multi-faceted browsing interfaces. In: Proceedings of the 2005 ACM Conference on Information and Knowledge Management, pp. 768–775 (2005)

12. Stoica, E., Hearst, M.A., Richardson, M.: Automatic creation of hierarchical faceted metadata structures. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (2007)
13. Klavans, J., Sheffield, C., Abels, E., Lin, J., Passonneau, R., Sidhu, T., Soergel, D.: Computational linguistics for metadata building (climb): using text mining for the automatic identification, categorization, and disambiguation of subject terms for image metadata. Multimedia Tools and Applications 42, 115–138 (2009), doi:10.1007/s11042-008-0253-9
14. Eklund, P., Goodall, P., Wray, T.: Folksonomy with practical taxonomy, a design for social metadata of the virtual museum of the pacific. In: The 6th International Conference on Information Technology and Applications, pp. 112–117. IEEE press, Los Alamitos (2009)
15. Grineva, M., Grinev, M., Lizorkin, D.: Extracting key terms from noisy and multitheme documents. In: Proceedings of the 18th International Conference on the World Wide Web (2009)
16. Kuznetsov, S., Obiedkov, S., Roth, C.: Reducing the representation complexity of lattice-based taxonomies. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 241–254. Springer, Heidelberg (2007)
17. Rorissa, A., Iyer, H.: Theories of cognition and image categorisation: what category labels reveal about basic level theory. Journal of the American Society for Information Science and Technology 9, 1383–1392 (2008)

Author Index