

Lecture Notes in Artificial Intelligence 2961

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Springer

Berlin

Heidelberg

New York

Hong Kong

London

Milan

Paris

Tokyo

Peter Eklund (Ed.)

Concept Lattices

Second International Conference
on Formal Concept Analysis, ICFCA 2004
Sydney, Australia, February 23-26, 2004
Proceedings



Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editor

Peter Eklund
The University of Wollongong
School of Information Technology and Computer Science
Northfields Avenue, Wollongong NSW 2522, Australia
E-mail: peklund@uow.edu.au

Cataloging-in-Publication Data applied for

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

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

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

ISSN 0302-9743

ISBN 3-540-21043-1 Springer-Verlag Berlin Heidelberg New York

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

Springer-Verlag is a part of Springer Science+Business Media
springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 10985977 06/3142 5 4 3 2 1 0

Preface

This volume contains the Proceedings of ICFCA 2004, the 2nd International Conference on Formal Concept Analysis. The ICFCA conference series aims to be the premier forum for the publication of advances in applied lattice and order theory and in particular scientific advances related to formal concept analysis.

Formal concept analysis emerged in the 1980s from efforts to restructure lattice theory to promote better communication between lattice theorists and potential users of lattice theory. Since then, the field has developed into a growing research area in its own right with a thriving theoretical community and an increasing number of applications in data and knowledge processing including data visualization, information retrieval, machine learning, data analysis and knowledge management.

In terms of theory, formal concept analysis has been extended into attribute exploration, Boolean judgment, contextual logic and so on to create a powerful general framework for knowledge representation and reasoning. This conference aims to unify theoretical and applied practitioners who use formal concept analysis, drawing on the fields of mathematics, computer and library sciences and software engineering. The theme of the 2004 conference was ‘Concept Lattices’ to acknowledge the colloquial term used for the line diagrams that appear in almost every paper in this volume.

ICFCA 2004 included tutorial sessions, demonstrating the practical benefits of formal concept analysis, and highlighted developments in the foundational theory and standards. The conference showcased the increasing variety of formal concept analysis software and included eight invited lectures from distinguished speakers in the field. Seven of the eight invited speakers submitted accompanying papers and these were reviewed and appear in this volume.

All regular papers appearing in this volume were refereed by at least two referees. In almost all cases three (or more) referee reports were returned. Long papers of approximately 14 pages represent substantial results deserving additional space based on the recommendations of reviewers. The final decision to accept the papers (as long, short or at all) was arbitrated by the Program Chair based on the referee reports. As Program Chair, I wish to thank the Program Committee and the additional reviewers for their involvement which ensured the high scientific quality of these proceedings. I also wish to particularly thank Prof. Paul Compton and Rudolf Wille, Dr. Richard Cole and Peter Becker for their support and enthusiasm.

January 2004

Peter Eklund

Organization

The International Conference on Formal Concept Analysis (ICFCA) is intended as a regular 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 Technical University of Darmstadt, Germany in 2003 and ICFCA 2004 in Sydney, Australia.

Executive Committee

Conference Chair	Paul Compton (University of NSW, Australia)
Program Chair	Peter Eklund (University of Wollongong, Australia)
Tutorials	Peter Becker and Bastian Wormuth
Workshops	Peter Becker

Program Committee

Claudio Carpineto (Fondazione Ugo Bordoni)
Richard Cole (University of Queensland)
Bernhard Ganter (Technical University Dresden)
Robert Godin (Université du Québec)
Rokia Missaoui (Université du Québec)
Engelbert Mephu Nguito (Université de Artois, Lens)
Uta Priss (Napier University)
Gregor Snelting (University of Passau)
Gerd Stumme (University of Karlsruhe)
Rudolf Wille (Darmstadt University of Technology)
Karl Erich Wolff (University of Applied Sciences, Darmstadt)

Additional Referees

P. Becker	J. Hereth Correia	S. Kuznetsov
F. Dau	J. Klinger	B. Wormuth
B. Davey		

Table of Contents

Preconcept Algebras and Generalized Double Boolean Algebras	1
<i>Rudolf Wille</i>	
Protoconcept Graphs: The Lattice of Conceptual Contents	14
<i>Joachim Hereth Correia and Julia Klinger</i>	
Signs and Formal Concepts	28
<i>Uta Priss</i>	
A Mathematical Model for TOSCANA-Systems: Conceptual Data Systems	39
<i>Joachim Hereth Correia and Tim B. Kaiser</i>	
BLID: An Application of Logical Information Systems to Bioinformatics	47
<i>Sébastien Ferré and Ross D. King</i>	
Formal Concept Analysis: Its Role in Teaching Lattice Theory	55
<i>Brian A. Davey</i>	
Concept Lattices for Information Visualization: Can Novices Read Line-Diagrams?	57
<i>Peter Eklund, Jon Ducrou, and Peter Brawn</i>	
Browsing Search Results via Formal Concept Analysis: Automatic Selection of Attributes	74
<i>Juan M. Cigarrán, Julio Gonzalo, Anselmo Peñas, and Felisa Verdejo</i>	
Formal Concept Analysis and Semantic File Systems	88
<i>Ben Martin</i>	
Numerical Analysis in Conceptual Systems with TOSCANAJ	96
<i>Peter Becker</i>	
Tool Support for FCA	104
<i>Thomas Tilley</i>	
Automated Lattice Drawing	112
<i>Ralph Freese</i>	
Congruences of Finite Distributive Concept Algebras	128
<i>Bernhard Ganter</i>	
When Is a Concept Algebra Boolean?	142
<i>Léonard Kwuida</i>	

Background Knowledge in Concept Graphs	156
<i>Frithjof Dau</i>	
Agreement Contexts in Formal Concept Analysis	172
<i>Richard Cole and Peter Becker</i>	
Towards a Conceptual Theory of Indistinguishable Objects	180
<i>Karl Erich Wolff</i>	
Conceptual Knowledge Processing with Formal Concept Analysis and Ontologies	189
<i>Philipp Cimiano, Andreas Hotho, Gerd Stumme, and Julien Tane</i>	
A First Step towards Protoconcept Exploration	208
<i>Björn Vormbrock</i>	
Geometry of Data Tables	222
<i>Tim B. Kaiser and Stefan E. Schmidt</i>	
Concept Extensions and Weak Clusters Associated with Multiway Dissimilarity Measures	236
<i>Jean Diatta</i>	
Unlocking the Semantics of Roget's Thesaurus Using Formal Concept Analysis	244
<i>L. John Old</i>	
FCA in Knowledge Technologies: Experiences and Opportunities	252
<i>Yannis Kalfoglou, Srinandan Dasmahapatra, and Yun-Heh Chen-Burger</i>	
Applying Formal Concept Analysis to Description Logics	261
<i>Franz Baader and Baris Sertkaya</i>	
Machine Learning and Formal Concept Analysis	287
<i>Sergei O. Kuznetsov</i>	
A Comparative Study of FCA-Based Supervised Classification Algorithms	313
<i>Huaiyu Fu, Huaiguo Fu, Patrik Njiwoua, and Engelbert Mephu Nguifo</i>	
Modelling Tacit Knowledge via Questionnaire Data	321
<i>Peter Busch and Debbie Richards</i>	
Predicate Invention and the Revision of First-Order Concept Lattices	329
<i>Michael Bain</i>	
The Power of Peircean Algebraic Logic (PAL)	337
<i>Joachim Hereth Correia and Reinhard Pöschel</i>	

Formal Concept Analysis for Knowledge Discovery and Data Mining: The New Challenges.....	352
<i>Petko Valtchev, Rokia Missaoui, and Robert Godin</i>	
AddIntent: A New Incremental Algorithm for Constructing Concept Lattices.....	372
<i>Dean van der Merwe, Sergei Obiedkov, and Derrick Kourie</i>	
QuDA: Applying Formal Concept Analysis in a Data Mining Environment	386
<i>Peter A. Grigoriev and Serhiy A. Yevtushenko</i>	
A Parallel Algorithm to Generate Formal Concepts for Large Data	394
<i>Huaiguo Fu and Engelbert Mephu Nguifo</i>	
Using Concept Lattices for Requirements Reconciliation	402
<i>Debbie Richards</i>	
Author Index	411

Preconcept Algebras and Generalized Double Boolean Algebras

Rudolf Wille

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D-64289 Darmstadt
wille@mathematik.tu-darmstadt.de

Abstract. *Boolean Concept Logic* as an integrated generalization of Contextual Object Logic and Contextual Attribute Logic can be substantially developed on the basis of preconcept algebras. The main results reported in this paper are the *Basic Theorem on Preconcept Algebras* and the Theorem characterizing the equational class generated by all preconcept algebras by the equational axioms of the *generalized double Boolean algebras*.

1 Preconcepts in Boolean Concept Logic

Concepts are the basic units of thought wherefore a concept-oriented mathematical logic is of great interest. G. Boole has offered the most influential foundation for such a logic which is based on a general conception of *signs* representing *classes* of objects from a given *universe of discourse* [Bo54]. In the language of Formal Concept Analysis [GW99a], Boole's basic notions can be explicated:

- for a “universe of discourse”, by the notion of a “*formal context*” defined as a mathematical structure $\mathbb{K} := (G, M, I)$ where G is a set whose elements are called “*objects*”, M is a set whose elements are called “*attributes*”, and I is a subset of $G \times M$ for which gIm (i.e. $(g, m) \in I$) is read: the object g *has* the attribute m ,
- for a “sign”, by the notion of an “*attribute*” of a formal context and,
- for a “class”, by the notion of an “*extent*” defined in a formal context $\mathbb{K} := (G, M, I)$ as a subset $Y' := \{g \in G \mid \forall m \in Y : gIm\}$ for some $Y \subseteq M$.

How Boole's logic of signs and classes may be developed as a *Contextual Attribute Logic* by means of Formal Concept Analysis is outlined in [GW99b]. The dual *Contextual Object Logic*, which is for instance used to determine conceptual contents of information [Wi03a], can be obtained from Contextual Attribute Logic by interchanging the role of objects and attributes so that, in particular, the notion of an “extent” is replaced by

- the notion of an “*intent*” defined in a formal context $\mathbb{K} := (G, M, I)$ as a subset $X' := \{m \in M \mid \forall g \in X : gIm\}$ for some $X \subseteq G$.

Since a concept, as a unit of thought, combines an extension consisting of objects and an intension consisting of attributes (properties, meanings) (cf. [Sch90], p.83ff.), a concept-oriented mathematical logic should be an integrated generalization of a mathematical attribute logic and a mathematical object logic. In our contextual approach based on Formal Concept Analysis, such an integrated generalization can be founded on

- the notion of a “*formal concept*” defined, in a formal context $\mathbb{K} := (G, M, I)$, as a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A = B'$, and $B = A'$ [Wi82],

and its generalizations:

- the notions of a “ \sqcap -*semiconcept*” (A, A') with $A \subseteq G$ and a “ \sqcup -*semiconcept*” (B', B) with $B \subseteq M$ [LW91], the notion of a “*protoconcept*” (A, B) with $A \subseteq G$, $B \subseteq M$, and $A'' = B'$ ($\Leftrightarrow B'' = A'$) [Wi00a], and the notion of a “*preconcept*” (A, B) with $A \subseteq G$, $B \subseteq M$, and $A \subseteq B'$ ($\Leftrightarrow B = A'$) [SW86].

Clearly, formal concepts are always semiconcepts, semiconcepts are always protoconcepts, and protoconcepts are always preconcepts. Since, for $X \subseteq G$ and $Y \subseteq M$, we always have $X''' = X'$ and $Y''' = Y'$, formal concepts can in general be constructed by forming (X'', X') or (Y', Y'') . The basic logical *derivations* $X \mapsto X'$ and $Y \mapsto Y'$ may be naturally generalized to the conceptual level by

- $(X, Y) \mapsto (X, X') \mapsto (X'', X')$ and $(X, Y) \mapsto (Y', Y) \mapsto (Y', Y'')$
for an arbitrary preconcept (X, Y) of $\mathbb{K} := (G, M, I)$.

It is relevant to assume that (X, Y) is a preconcept because otherwise we would obtain $Y \not\subseteq X'$ and $X \not\subseteq Y'$, i.e., (X, X') and (Y', Y) would not be extensions of (X, Y) with respect to the order \subseteq^2 defined by

- $(X_1, Y_1) \subseteq^2 (X_2, Y_2) : \Leftrightarrow X_1 \subseteq X_2 \text{ and } Y_1 \subseteq Y_2$.

Notice that, in the ordered set $(\mathfrak{V}(\mathbb{K}), \subseteq^2)$ of all preconcepts of a formal context $\mathbb{K} := (G, M, I)$, the formal concepts of \mathbb{K} are exactly the maximal elements and the protoconcepts of \mathbb{K} are just the elements which are below exactly one maximal element (formal concept).

For contextually developing a *Boolean Concept Logic* as an integrated generalization of the Contextual Object Logic and the Contextual Attribute Logic, *Boolean operations* have to be introduced on the set $\mathfrak{V}(\mathbb{K})$ of all preconcepts of a formal context $\mathbb{K} := (G, M, I)$. That shall be done in the same way as for semiconcepts [LW91] and for protoconcepts [Wi00a]:

$$\begin{aligned}
 (A_1, B_1) \sqcap (A_2, B_2) &:= (A_1 \cap A_2, (A_1 \cap A_2)') \\
 (A_1, B_1) \sqcup (A_2, B_2) &:= ((B_1 \cap B_2)', B_1 \cap B_2) \\
 \neg(A, B) &:= (G \setminus A, (G \setminus A)') \\
 \neg(A, B) &:= ((M \setminus B)', M \setminus B) \\
 \perp &:= (\emptyset, M) \\
 \top &:= (G, \emptyset)
 \end{aligned}$$

The set $\mathfrak{B}(\mathbb{K})$ together with the operations $\sqcap, \sqcup, \neg, \neg\!, \perp$, and \top is called the *preconcept algebra* of \mathbb{K} and is denoted by $\underline{\mathfrak{B}}(\mathbb{K})$; the operations are named “*meet*”, “*join*”, “*negation*”, “*opposition*”, “*nothing*”, and “*all*”. For the structural analysis of the preconcept algebra $\underline{\mathfrak{B}}(\mathbb{K})$, it is useful to define additional operations on $\mathfrak{B}(\mathbb{K})$:

$$\begin{aligned} \mathfrak{a} \sqcup \mathfrak{b} &:= \neg(\neg\mathfrak{a} \sqcap \neg\mathfrak{b}) \text{ and } \mathfrak{a} \sqcap \mathfrak{b} := \neg(\neg\mathfrak{a} \sqcup \neg\mathfrak{b}), \\ \top &:= \neg\perp \text{ and } \perp := \neg\top. \end{aligned}$$

The semiconcepts resp. protoconcepts of \mathbb{K} form subalgebras $\underline{\mathfrak{H}}(\mathbb{K})$ resp. $\underline{\mathfrak{P}}(\mathbb{K})$ of $\underline{\mathfrak{B}}(\mathbb{K})$ which are called the *semiconcept algebra* resp. *protoconcept algebra* of \mathbb{K} . The set $\mathfrak{H}_{\sqcap}(\mathbb{K})$ of all \sqcap -semiconcepts is closed under the operations $\sqcap, \sqcup, \neg, \perp$, and \top ; therefore, $\underline{\mathfrak{H}}_{\sqcap}(\mathbb{K}) := (\mathfrak{H}_{\sqcap}(\mathbb{K}), \sqcap, \sqcup, \neg, \perp, \top)$ is a Boolean algebra isomorphic to the Boolean algebra of all subsets of G . Dually, the set $\mathfrak{H}_{\sqcup}(\mathbb{K})$ of all \sqcup -semiconcepts is closed under the operations $\sqcap, \sqcup, \neg, \perp$, and \top ; therefore, $\underline{\mathfrak{H}}_{\sqcup}(\mathbb{K}) := (\mathfrak{H}_{\sqcup}(\mathbb{K}), \sqcap, \sqcup, \neg, \perp, \top)$ is a Boolean algebra antiisomorphic to the Boolean algebra of all subsets of M . Furthermore, $\mathfrak{B}(\mathbb{K}) = \mathfrak{H}_{\sqcap}(\mathbb{K}) \cap \mathfrak{H}_{\sqcup}(\mathbb{K})$, and $(\mathfrak{B}(\mathbb{K}), \wedge, \vee)$ is the so-called *concept lattice* of \mathbb{K} with the operations \wedge and \vee induced by the operations \sqcap and \sqcup , respectively. The general order relation \sqsubseteq of $\underline{\mathfrak{B}}(\mathbb{K})$, which coincides on $\mathfrak{B}(\mathbb{K})$ with the subconcept-superconcept-order \leq , is defined by

$$(A_1, B_1) \sqsubseteq (A_2, B_2) :\iff A_1 \subseteq A_2 \text{ and } B_1 \supseteq B_2.$$

The introduced notions found a Boolean Concept Logic in which the Contextual Object Logic and the Contextual Attribute Logic can be integrated by transforming any object sets X to the \sqcap -semiconcept (X, X') and any attribute set Y to the corresponding \sqcup -semiconcept (Y', Y) . In the case of Contextual Attribute Logic [GW99b], this integration comprises a transformation of the Boolean compositions of attributes which is generated by the following elementary assignments:

$$\begin{aligned} m \wedge n &\mapsto (\{m\}', \{m\}) \sqcap (\{n\}', \{n\}), \\ m \vee n &\mapsto (\{m\}', \{m\}) \sqcup (\{n\}', \{n\}), \\ \neg m &\mapsto \neg(\{m\}', \{m\}). \end{aligned}$$

In the dual case of Contextual Object Logic, the corresponding transformation uses the operations \sqcap, \sqcup , and \neg .

Preconcept algebras can be illustrated by *line diagrams* which shall be demonstrated by using the small formal context in Fig.1. The line diagram of the preconcept algebra of that formal context is shown in Fig.2: the formal concepts are represented by the black circles, the proper \sqcap -semiconcepts by the circles with only a black lower half, the proper \sqcup -semiconcepts by the circles with only a black upper half, and the proper preconcepts (which are even not protoconcepts) by the unblackened circles. An object (attribute) belongs to a preconcept if and only if its name is attached to a circle representing a subpreconcept (superpreconcept) of that preconcept. The regularity of the line diagram in Fig.2 has a general reason which becomes clear by the following proposition:

	male	female	old	young
father	×		×	
mother		×	×	
son	×			×
daughter		×		×

Fig. 1. A context \mathbb{K}^f of family members

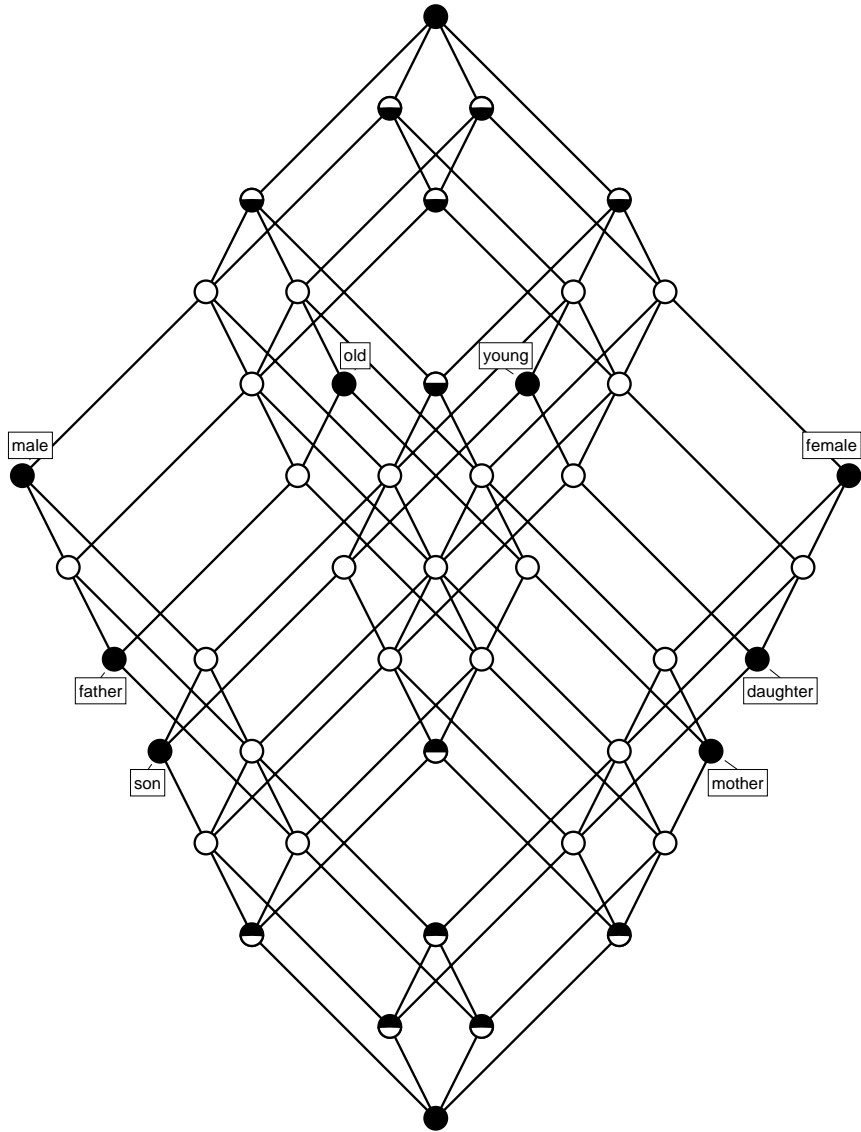


Fig. 2. Line diagram of the preconcept algebra of the formal context \mathbb{K}^f in Fig.1

Proposition 1 For a formal context $\mathbb{K} := (G, M, I)$, the ordered set $(\mathfrak{V}(\mathbb{K}), \sqsubseteq)$ is a completely distributive complete lattice, which is isomorphic to the concept lattice of the formal context $\mathbb{V}(\mathbb{K}) := (G \dot{\cup} M, G \dot{\cup} M, I \cup (\neq \setminus G \times M))$.

Proof For $(A_t, B_t) \in \mathfrak{V}(\mathbb{K})$ ($t \in T$), we obviously have

$$\inf_{t \in T} (A_t, B_t) = \left(\bigcap_{t \in T} A_t, \bigcup_{t \in T} B_t \right) \text{ and } \sup_{t \in T} (A_t, B_t) = \left(\bigcup_{t \in T} A_t, \bigcap_{t \in T} B_t \right);$$

hence $\mathfrak{V}(\mathbb{K})$ is a complete sublattice of the completely distributive complete lattice $(\mathfrak{P}(G), \sqsubseteq) \times (\mathfrak{P}(M), \supseteq)$ which proves the first assertion. For proving the second assertion, we consider the assignment $(A, B) \mapsto (A \cup (M \setminus B), (G \setminus A) \cup B)$. It can be easily checked that $(A \cup (M \setminus B), (G \setminus A) \cup B)$ is a formal concept of $\mathbb{V}(\mathbb{K})$. Let (C, D) be an arbitrary formal concept of $\mathbb{V}(\mathbb{K})$. Obviously, $C = (C \cap G) \cup (M \setminus D)$, $D = (G \setminus C) \cup (D \cap M)$, and $(C \cap G, D \cap M)$ is a preconcept of \mathbb{K} . Therefore we obtain $(C \cap G, D \cap M) \mapsto (C, D)$. Thus, ι is a bijection from $\mathfrak{V}(\mathbb{K})$ onto $\mathfrak{B}(\mathbb{V}(\mathbb{K}))$. Since $(A_1, B_1) \sqsubseteq (A_2, B_2) \iff A_1 \subseteq A_2 \text{ and } B_1 \supseteq B_2 \iff A_1 \cup (M \setminus B_1) \subseteq A_2 \cup (M \setminus B_2) \iff (A_1 \cup (M \setminus B_1), (G \setminus A_1) \cup B_1) \leq (A_2 \cup (M \setminus B_2), (G \setminus A_2) \cup B_2)$, the bijection ι is even an isomorphism from $\mathfrak{V}(\mathbb{K})$ onto $\mathfrak{B}(\mathbb{V}(\mathbb{K}))$. \square

2 The Basic Theorem on Preconcept Algebras

A detailed understanding of the *structure of preconcept algebras* is basic for Boolean Concept Logic. To start the necessary structure analysis, we first determine basic equations valid in all preconcept algebras and study abstractly the class of all algebras satisfying those equations. The aim is to prove a characterization of preconcept algebras analogously to the *Basic Theorem on Concept Lattices* [Wi82].

Proposition 2 In a preconcept algebra $\mathfrak{A}(\mathbb{K})$ the following equations are valid:

1a) $(x \sqcap x) \sqcap y = x \sqcap y$	1b) $(x \sqcup x) \sqcup y = x \sqcup y$
2a) $x \sqcap y = y \sqcap x$	2b) $x \sqcup y = y \sqcup x$
3a) $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$	3b) $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$
4a) $x \sqcap (x \sqcup y) = x \sqcap x$	4b) $x \sqcup (x \sqcap y) = x \sqcup x$
5a) $x \sqcap (x \sqcup y) = x \sqcap x$	5b) $x \sqcup (x \sqcap y) = x \sqcup x$
6a) $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$	6b) $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$
7a) $\neg \neg (x \sqcap y) = x \sqcap y$	7b) $\neg \neg (x \sqcup y) = x \sqcup y$
8a) $\neg (x \sqcap x) = \neg x$	8b) $\neg (x \sqcup x) = \neg x$
9a) $x \sqcap \neg x = \perp$	9b) $x \sqcup \neg x = \top$
10a) $\neg \perp = \top \sqcap \top$	10b) $\neg \top = \perp \sqcup \perp$
11a) $\neg \top = \perp$	11b) $\neg \perp = \top$
12a) $x \sqcap \sqcup = x \sqcup$	12b) $x \sqcup \sqcap = x \sqcap$

where $t_{\sqcap} := t \sqcap t$ and $t_{\sqcup} := t \sqcup t$ is defined for every term t .

Proof The equations of the proposition can be easily verified in preconcept algebras. This shall only be demonstrated by proving 4a) and 12a): $(A, B) \sqcap ((A, B) \sqcup (C, D)) = (A, B) \sqcap ((B \cap D)', B \cap D) = (A \cap (B \cap D)', (A \cap (B \cap D))') = (A, A') = (A, B) \sqcap (A, B)$ and $(A, B) \sqcap \sqcup = (A, A') \sqcup = (A'', A') \sqcap = (A'', A') = (A, A') \sqcup = (A, B) \sqcup$. \square

An algebra $\underline{D} := (D, \sqcap, \sqcup, \neg, \neg, \perp, \top)$ of type $(2, 2, 1, 1, 0, 0)$ is called a *weak double Boolean algebra* if it satisfies the equations 1a) to 11a) and 1b) to 11b) and a *generalized double Boolean algebra* if it satisfies the equations 1a) to 12a) and 1b) to 12b) of Proposition 2; if, in addition, it even satisfies the equation 12) $x_{\sqcap \sqcup} = x_{\sqcup \sqcap}$, \underline{D} is called a *double Boolean algebra* [HLSW00]. For weak double Boolean algebras, further operations are defined as in the case of preconcept algebras:

$$\begin{aligned} x \sqcup y &:= \neg(\neg x \sqcap \neg y) \text{ and } x \sqcap y := \neg(\neg x \sqcup \neg y), \\ \top &:= \neg \perp \text{ and } \perp := \neg \top, \\ x_{\sqcap} &:= x \sqcap x \text{ and } x_{\sqcup} := x \sqcup x. \end{aligned}$$

By Proposition 2, each preconcept algebra is a generalized double Boolean algebra. Protoconcept algebras are double Boolean algebras. Semiconcepts algebras satisfy the additional condition 13) $x \sqcap x = x$ or $x \sqcup x = x$. A weak double Boolean algebra \underline{D} satisfying the condition 13) is called *pure*, because it is only the union of the two subsets $D_{\sqcap} := \{x \in D \mid x \sqcap x = x\}$ and $D_{\sqcup} := \{x \in D \mid x \sqcup x = x\}$ which both carry a Boolean structure, i.e., $\underline{D}_{\sqcap} := (D_{\sqcap}, \sqcap, \sqcup, \neg, \perp, \top)$ and $\underline{D}_{\sqcup} := (D_{\sqcup}, \sqcap, \sqcup, \neg, \perp, \top)$ are Boolean algebras. As on preconcept algebras, a quasiorder is defined on weak double Boolean algebras by

$$x \sqsubseteq y : \iff x \sqcap y = x \sqcap x \text{ and } x \sqcup y = y \sqcup y.$$

A weak double Boolean algebra \underline{D} is said to be *complete* if the Boolean algebras \underline{D}_{\sqcap} and \underline{D}_{\sqcup} are complete. The existing infimum resp. supremum of a subset A of D_{\sqcap} are denoted by $\bigcap A$ resp. $\bigcup A$ and, dually, of a subset B of D_{\sqcup} by $\bigcap B$ resp. $\bigcup B$. In general, it is appropriate to define the arbitrary meet and join in \underline{D} by $\bigcap C := \bigcap \{c_{\sqcap} \mid c \in C\}$ and $\bigcup C := \bigcup \{c_{\sqcup} \mid c \in C\}$ for arbitrary subsets C of D . Clearly, preconcept algebras are examples of complete double Boolean algebras.

In [HLSW00], weak double Boolean algebras $\underline{D} := (D, \sqcap, \sqcup, \neg, \neg, \perp, \top)$ with the quasiorder \sqsubseteq defined by $x \sqsubseteq y : \iff (x \sqcap y = x \sqcap x \text{ and } x \sqcup y = y \sqcup y)$ are characterized as quasiordered sets (D, \sqsubseteq) satisfying the following conditions:

1. (D, \sqsubseteq) has a smallest element 0 and a largest element 1;
2. there is a subset D_{\sqcap} of D so that $(D_{\sqcap}, \sqsubseteq)$ is a Boolean algebra whose operations coincide with the operations $\sqcap, \sqcup, \neg, \perp$, and \top of \underline{D} ;
3. there is a subset D_{\sqcup} of D so that $(D_{\sqcup}, \sqsubseteq)$ is a Boolean algebra whose operations coincide with the operations $\sqcap, \sqcup, \neg, \perp$, and \top of \underline{D} ;
4. for $x \in D$, there exists $x_{\sqcap} \in D_{\sqcap}$ with $y \sqsubseteq x_{\sqcap} \iff y \sqsubseteq x$ for all $y \in D_{\sqcap}$;
5. for $x \in D$, there exists $x_{\sqcup} \in D_{\sqcup}$ with $y \sqsupseteq x_{\sqcup} \iff y \sqsupseteq x$ for all $y \in D_{\sqcup}$;
6. $x \sqsubseteq y \iff x_{\sqcap} \sqsubseteq y_{\sqcap}$ and $x_{\sqcup} \sqsubseteq y_{\sqcup}$.

The links between the algebraically and order-theoretically given operations are established by the following equations: $x \sqcap y = x_{\sqcap} \sqcap y_{\sqcap}$, $x \sqcup y = x_{\sqcup} \sqcup y_{\sqcup}$, $\neg x = \neg(x_{\sqcap})$, $\neg x = \neg(x_{\sqcup})$, $\perp = 0$, and $\top = 1$. Therefore, for an order-theoretic characterization of generalized double Boolean algebras one has to add the further condition

7. $x_{\sqcap \sqcap} = x_{\sqcap}$ and $x_{\sqcup \sqcup} = x_{\sqcup}$.

For formulating the *Basic Theorem on Preconcept Algebras*, we adapt notions which have been used for proving the corresponding *Basic Theorem on Protoconcept Algebras* [VW03]: A weak double Boolean algebra \underline{D} is called *contextual* if its quasiorder \sqsubseteq is antisymmetric. A contextual weak double Boolean algebra \underline{D} is said to be *totally contextual* if, in addition, for each $x \in \underline{D}_\sqcap$ and $y \in \underline{D}_\sqcup$ with $x_\sqcup \sqsubseteq y_\sqcap$ there is a unique $z \in \underline{D}$ with $z_\sqcap = x$ and $z_\sqcup = y$.

Theorem 1 (The Basic Theorem on Preconcept Algebras) *For a context $\mathbb{K} := (G, M, I)$, the preconcept algebra $\mathfrak{V}(\mathbb{K})$ of \mathbb{K} is a complete totally contextual generalized double Boolean algebra whose Boolean algebras $\mathfrak{H}_\sqcap(\mathbb{K})$ and $\mathfrak{H}_\sqcup(\mathbb{K})$ are atomic. The (arbitrary) meet and join of $\mathfrak{V}(\mathbb{K})$ are given by*

$$\bigsqcap_{t \in T} (A_t, B_t) = \left(\bigcap_{t \in T} A_t, \left(\bigcap_{t \in T} A_t \right)' \right) \quad \text{and} \quad \bigsqcup_{t \in T} (A_t, B_t) = \left(\left(\bigcap_{t \in T} B_t \right)', \bigcap_{t \in T} B_t \right).$$

In general, a complete totally contextual generalized double Boolean algebra \underline{D} whose Boolean algebras \underline{D}_\sqcap and \underline{D}_\sqcup are atomic, is isomorphic to $\mathfrak{V}(\mathbb{K})$ if and only if there exist a bijection $\tilde{\gamma}$ from G onto the set $\mathcal{A}(\underline{D}_\sqcap)$ of all atoms of \underline{D}_\sqcap and a bijection $\tilde{\mu}$ from M onto the set $\mathcal{C}(\underline{D}_\sqcup)$ of all coatoms of \underline{D}_\sqcup such that $gIm \iff \tilde{\gamma}(g) \sqsubseteq \tilde{\mu}(m)$ for all $g \in G$ and $m \in M$. In particular, for any complete totally contextual generalized double Boolean algebra \underline{D} whose Boolean algebras are atomic, we get $\underline{D} \cong \mathfrak{V}(\mathcal{A}(\underline{D}_\sqcap), \mathcal{C}(\underline{D}_\sqcup), \sqsubseteq)$, i.e., the preconcept algebras are up to isomorphism the complete totally contextual generalized double Boolean algebras \underline{D} whose Boolean algebras \underline{D}_\sqcap and \underline{D}_\sqcup are atomic.

Proof Using Proposition 2, it is straightforward to check that every preconcept algebra $\mathfrak{V}(\mathbb{K})$ is a complete contextual generalized double Boolean algebra whose Boolean algebras $\mathfrak{H}_\sqcap(\mathbb{K})$ and $\mathfrak{H}_\sqcup(\mathbb{K})$ are atomic. For semiconcepts (A, A') and (B', B) with $(A'', A') = (A, A')_\sqcup \sqsubseteq (B', B)_\sqcap = (B', B'')$, (A, B) is the unique preconcept with $(A, B)_\sqcap = (A, A')$ and $(A, B)_\sqcup = (B', B)$; hence $\mathfrak{V}(\mathbb{K})$ is even totally contextual. Because of $(A, B)_\sqcap = (A, A')$ and $(A, B)_\sqcup = (B', B)$ for each preconcept (A, B) , we obtain

$$\begin{aligned} \bigsqcap_{t \in T} (A_t, B_t) &= \inf_{\mathfrak{H}_\sqcap(\mathbb{K})} \{ (A_t, A'_t) \mid t \in T \} = \left(\bigcap_{t \in T} A_t, \left(\bigcap_{t \in T} A_t \right)' \right), \\ \bigsqcup_{t \in T} (A_t, B_t) &= \sup_{\mathfrak{H}_\sqcup(\mathbb{K})} \{ (B'_t, B_t) \mid t \in T \} = \left(\left(\bigcap_{t \in T} B_t \right)', \bigcap_{t \in T} B_t \right). \end{aligned}$$

Now, let $\varphi : \mathfrak{V}(\mathbb{K}) \rightarrow \underline{D}$ be an isomorphism. Then the desired bijections are defined by $\tilde{\gamma}(g) := \varphi(\{g\}, \{g\}')$ and $\tilde{\mu}(m) := \varphi(\{m\}', \{m\})$. Since $\mathcal{A}(\mathfrak{H}_\sqcap(\mathbb{K})) = \{(\{g\}, \{g\}') \mid g \in G\}$ and $\mathcal{C}(\mathfrak{H}_\sqcup(\mathbb{K})) = \{(\{m\}', \{m\}) \mid m \in M\}$, it follows $\mathcal{A}(\underline{D}_\sqcap(\mathbb{K})) = \{\tilde{\gamma}(g) \mid g \in G\}$ and $\mathcal{C}(\underline{D}_\sqcup(\mathbb{K})) = \{\tilde{\mu}(m) \mid m \in M\}$. Thus, $\tilde{\gamma}$ is indeed a bijection from G onto $\mathcal{A}(\underline{D}_\sqcap)$ and $\tilde{\mu}$ a bijection from M onto $\mathcal{C}(\underline{D}_\sqcup)$. Furthermore, $gIm \iff (g \in \{m\}' \text{ and } m \in \{g\}') \iff (\{g\}, \{g\}') \sqsubseteq (\{m\}', \{m\}) \iff \tilde{\gamma}(g) \sqsubseteq \tilde{\mu}(m)$ for all $g \in G$ and $m \in M$.

Conversely, we assume the existence of the bijections $\tilde{\gamma}$ and $\tilde{\mu}$ with the required properties. Then we define two maps $\varphi_\sqcap : \mathfrak{H}_\sqcap(\mathbb{K}) \rightarrow \underline{D}_\sqcap$ and $\varphi_\sqcup :$

$\underline{\mathfrak{H}}_{\sqcup}(\mathbb{K}) \rightarrow \underline{D}_{\sqcup}$ by $\varphi_{\sqcap}(A, A') := \bigsqcup \{\tilde{\gamma}(g) \mid g \in A\}$ and $\varphi_{\sqcup}(B', B) := \bigsqcap \{\tilde{\mu}(m) \mid m \in B\}$. Since \underline{D}_{\sqcap} and \underline{D}_{\sqcup} are complete atomic Boolean algebras, φ_{\sqcap} and φ_{\sqcup} are isomorphisms onto those Boolean algebras. For an arbitrary \sqcap -semiconcept (A, A') of \mathbb{K} , let $x := \varphi_{\sqcap}(A, A')$ and $y := \varphi_{\sqcup}(A'', A')$. Condition 4 (listed above) yields that $x_{\sqcup} = \bigsqcap \{b \in \underline{D}_{\sqcup} \mid x \sqsubseteq b\} = \bigsqcap \{c \in \mathcal{C}(\underline{\mathfrak{H}}_{\sqcup}(\mathbb{K})) \mid x \sqsubseteq c\} = y$ because of the equivalence $gIm \iff \tilde{\gamma}(g) \sqsubseteq \tilde{\mu}(m)$. Thus,

$$(\varphi_{\sqcap}(A, A'))_{\sqcup} = \varphi_{\sqcup}((A, A')_{\sqcup}) \text{ and } (\varphi_{\sqcup}(B', B))_{\sqcap} = \varphi_{\sqcap}((B', B)_{\sqcap})$$

because, for a \sqcup -semiconcept (B', B) of \mathbb{K} , we obtain dually $y_{\sqcap} = x$ if $y := \varphi_{\sqcup}(B', B)$ and $x := \varphi_{\sqcap}(B', B'')$. If (A, B) is even a formal concept of \mathbb{K} , we have $x = y_{\sqcap} = x_{\sqcup\sqcap} = x_{\sqcap\sqcup\sqcap} = x_{\sqcap\sqcup} \in D_{\sqcap} \cap D_{\sqcup}$ by the equation 12a) in Proposition 2 (in the analogous proof for semiconcept algebras in [VW03] we could just directly apply the equation 12). Now it follows that $\varphi_{\sqcap}(A, B) = x = x_{\sqcup} = y = \varphi_{\sqcup}(A, B)$. Thus, φ_{\sqcap} and φ_{\sqcup} coincide on $\mathfrak{B}(\mathbb{K}) (= \mathfrak{H}(\mathbb{K})_{\sqcap} \cap \mathfrak{H}(\mathbb{K})_{\sqcup})$ and therefore $\varphi(A, A') := \varphi_{\sqcap}(A, A')$ for $A \subseteq G$ and $\varphi(B', B) := \varphi_{\sqcup}(B', B)$ for $B \subseteq M$ defines a bijection φ from $\underline{\mathfrak{H}}(\mathbb{K})$ onto \underline{D}_p .

Since \underline{D} is totally contextual, φ can be uniquely extended to a bijection $\hat{\varphi}$ from $\underline{\mathfrak{B}}(\mathbb{K})$ onto \underline{D} : For a preconcept (A, B) which is not a semiconcept and for $x := \varphi(A, A')$ and $y := \varphi(B', B)$, we obtain $x_{\sqcup} \sqsubseteq y_{\sqcap}$; hence there is a unique $z(A, B) \in \underline{D}$ with $z(A, B)_{\sqcap} = x$ and $z(A, B)_{\sqcup} = y$. Thus, we can indeed extend φ to a bijection $\hat{\varphi} : \underline{\mathfrak{B}}(\mathbb{K}) \rightarrow \underline{D}$ with $\hat{\varphi}(A, B) = z(A, B)$ for all preconcepts (A, B) which are not semiconcepts.

$\hat{\varphi}$ preserves the operations of generalized double Boolean algebras which can be seen as follows: Since φ_{\sqcap} and φ_{\sqcup} are isomorphisms between the corresponding Boolean algebras and since $(\hat{\varphi}(A, B))_{\sqcap} = \varphi_{\sqcap}(A, A') = \hat{\varphi}((A, B)_{\sqcap})$, we get

$$\begin{aligned} \hat{\varphi} \bigsqcap_{t \in T} (A_t, B_t) &= \varphi_{\sqcap} \bigsqcap_{t \in T} (A_t, B_t)_{\sqcap} = \bigsqcap_{t \in T} \varphi_{\sqcap}((A_t, B_t)_{\sqcap}) \\ &= \bigsqcap_{t \in T} (\hat{\varphi}(A_t, B_t))_{\sqcap} = \bigsqcap_{t \in T} (\hat{\varphi}(A_t, B_t)); \\ \hat{\varphi}(\neg(A, B)) &= \varphi_{\sqcap}(\neg((A, B)_{\sqcap})) = \neg(\varphi_{\sqcap}((A, B)_{\sqcap})) \\ &= \neg((\hat{\varphi}(A, B))_{\sqcap}) = \neg(\hat{\varphi}(A, B)); \\ \hat{\varphi}(\emptyset, M) &= \varphi_{\sqcap}(\emptyset, M) = \perp. \end{aligned}$$

Dually, we obtain that $\hat{\varphi}$ preserves joins \bigsqcup , opposition \neg , and the top element \top .

Finally, let \underline{D} be a complete totally contextual generalized double Boolean algebra whose Boolean algebras are atomic, let $\tilde{\gamma}$ be the identity on $\mathcal{A}(\underline{D}_{\sqcap})$, and let $\tilde{\mu}$ be the identity on $\mathcal{C}(\underline{D}_{\sqcup})$. Then the already proved second part of the basic theorem yields directly the claimed isomorphism $\underline{D} \cong \underline{\mathfrak{B}}(\mathcal{A}(\underline{D}_{\sqcap}), \mathcal{C}(\underline{D}_{\sqcup}), \sqsubseteq)$. \square

The Basic Theorem on Preconcept Algebras may be used to check the correctness of a *line diagram representation* of a preconcept algebra. How this is performed shall be demonstrated by the formal context $\mathbb{K}^f := (G^f, M^f, I^f)$ in Fig.1 and its corresponding line diagram in Fig.2. In this figure,

- the preconcept algebra $\underline{\mathfrak{M}}(\mathbb{K}^f)$ is first of all drawn as an ordered set (V, \sqsubseteq) with 0 and 1;
- the 16 circles with black lower half represent a Boolean algebra $(V_{\sqcap}, \sqsubseteq)$ which is isomorphic to $\underline{\mathfrak{H}}_{\sqcap}(\mathbb{K}^f)$;
- the 16 circles with black upper half represent a Boolean algebra $(V_{\sqcup}, \sqsubseteq)$ which is isomorphic to $\underline{\mathfrak{H}}_{\sqcup}(\mathbb{K}^f)$;
- the conditions 4., 5., 6., and 7. of the above order-theoretic characterization of generalized double Boolean algebras are satisfied;
- the bijection $\tilde{\gamma}$ is indicated by the attachment of the object names to the represented atoms of the Boolean algebra $\underline{\mathfrak{H}}_{\sqcap}(\mathbb{K}^f)$;
- the bijection $\tilde{\mu}$ is indicated by the attachment of the attribute names to the represented coatoms of the Boolean algebra $\underline{\mathfrak{H}}_{\sqcup}(\mathbb{K}^f)$;
- finally, there is an ascending path of line segments from a circle with an object label to a circle with an attribute label if and only if the object has the attribute according to the formal context \mathbb{K}^f .

After checking all of this, we know by the Basic Theorem on Preconcept Algebras that the labelled line diagram in Fig.2 correctly represents the semiconcept algebra of the formal context \mathbb{K}^f given in Fig.1. An alternative for checking the correctness of a line diagram representation of a preconcept algebra can be based on Proposition 1 and the Basic Theorem on Concept Lattices (see [GW99a], p.20).

3 The Equational Class Generated by all Preconcept Algebras

An important question is whether the equations of Proposition 2 are enough for determining the *equational class generated by all preconcept algebras*, i.e., whether each equation valid in all preconcept algebras can be entailed by the equations of Proposition 2. This can indeed be proved, but it needs further investigations of generalized double Boolean algebras. The following four lemmas are adapted from the analogous investigations of double Boolean algebras in [Wi00a].

Lemma 1 *In a weak double Boolean algebra the following conditions are satisfied:*

- (1) $x \sqcap y \sqsubseteq x \sqsubseteq x \sqcup y$,
- (2) *the mapping $x \rightarrow x \sqcap y$ preserves \sqsubseteq and \sqcap ,*
- (3) *the mapping $x \rightarrow x \sqcup y$ preserves \sqsubseteq and \sqcup .*

Proof (1): 2a), 3a), and 1a) yield $(x \sqcap y) \sqcap x = y \sqcap (x \sqcap x) = y \sqcap (y \sqcap (x \sqcap x)) = (x \sqcap y) \sqcap (x \sqcap y)$. By 2b) and 4b) it follows $(x \sqcap y) \sqcup x = x \sqcup x$. Hence $x \sqcap y \sqsubseteq x$. Dually, we obtain $x \sqsubseteq x \sqcup y$. (2) and (3) follow straight forward. Let us only mention that $x \sqsubseteq y$ implies $(x \sqcap z) \sqcup (y \sqcap z) = (x \sqcap x \sqcap z) \sqcup (y \sqcap z) = (x \sqcap y \sqcap z) \sqcup (y \sqcap z) = (y \sqcap z) \sqcup (y \sqcap z)$ by 4b). \square

For proving that each generalized double Boolean algebra can be quasi-embedded into the preconcept algebra of a suitable context, the notion of a filter and an ideal of a weak double Boolean algebra \underline{D} is defined: A subset F of \underline{D} is called a *filter* if $x \in F$ and $x \sqsubseteq y$ in \underline{D} imply $y \in F$ and if $x \in F$ and $y \in F$ imply $x \sqcap y \in F$; an *ideal* of \underline{D} is dually defined. A subset F_0 is called a *base* of the filter F if $F = \{y \in D \mid x \sqsubseteq y \text{ for some } x \in F_0\}$; again, a base of an ideal is dually defined.

Lemma 2 *Let F be a filter of a weak double Boolean algebra \underline{D} .*

- (1) *$F \cap D_\sqcap$ and $F \cap D_\sqcup$ are filters of the Boolean algebras \underline{D}_\sqcap and \underline{D}_\sqcup , respectively.*
- (2) *Each filter of the Boolean algebra \underline{D}_\sqcap is a base of some filter of \underline{D} ; in particular, $F \cap D_\sqcap$ is a base of F .*

Proof Since the restrictions of \sqcap and \sqsubseteq to D_\sqcap are the meet operation and the order relation of the Boolean algebra \underline{D}_\sqcap , $F \cap D_\sqcap$ is obviously a filter of \underline{D}_\sqcap . $F \cap D_\sqcup$ is a filter of the Boolean algebra \underline{D}_\sqcup because $x \sqcap y \sqsubseteq x \sqcap y$ for arbitrary $x, y \in D_\sqcup$, namely $x \sqcap y$ is a lower bound of x and y by Lemma 1(1) and $x \sqcap y$ is the greatest lower bound of x and y since the restriction of \sqsubseteq to D_\sqcup is the order of the Boolean algebra \underline{D}_\sqcup . Now, let E be a filter of \underline{D}_\sqcap . For $x_1 \sqsubseteq y_1$ and $x_2 \sqsubseteq y_2$ with $x_1, x_2 \in E$, we obtain $x_1 \sqcap x_2 \sqsubseteq x_1 \sqcap y_2 \sqsubseteq y_1 \sqcap y_2$ by Lemma 1(2); hence $\{y \in D \mid x \sqsubseteq y \text{ for some } x \in E\}$ is a filter in \underline{D} with E as a base. For $y \in F$ we have that $y \sqcap y \in F \cap D_\sqcap$ and $y \sqcap y \sqsubseteq y$ by Lemma 1(1). Thus, $F \cap D_\sqcap$ is a base of F . \square

Let $\mathfrak{F}_p(\underline{D})$ be the set of all filters F of the weak double Boolean algebra \underline{D} for which $F \cap D_\sqcap$ is a prime filter of the Boolean algebra \underline{D}_\sqcap , and let $\mathfrak{I}_p(\underline{D})$ be the set of all ideals I of \underline{D} for which $I \cap D_\sqcup$ is a prime ideal of the Boolean algebra \underline{D}_\sqcup (for the definitions and results concerning prime filters and prime ideals see [DP92], p.185ff.). Now, we define the *standard context* for a weak double Boolean algebra \underline{D} by

$$\mathbb{K}(\underline{D}) := (\mathfrak{F}_p(\underline{D}), \mathfrak{I}_p(\underline{D}), \Delta)$$

where $F \Delta I :\Leftrightarrow F \cap I \neq \emptyset$ for $F \in \mathfrak{F}_p(\underline{D})$ and $I \in \mathfrak{I}_p(\underline{D})$. For $x \in D$, let $\mathfrak{F}_x := \{F \in \mathfrak{F}_p(\underline{D}) \mid x \in F\}$ and let $\mathfrak{I}_x := \{I \in \mathfrak{I}_p(\underline{D}) \mid x \in I\}$.

Lemma 3 *The derivations in $\mathbb{K}(\underline{D})$ yield:*

- (1) $\mathfrak{F}'_x = \mathfrak{I}_x = \mathfrak{I}_{x_\sqcup}$ for all $x \in D_\sqcap$,
- (2) $\mathfrak{I}'_y = \mathfrak{F}_y = \mathfrak{F}_{y_\sqcap}$ for all $y \in D_\sqcup$,

Proof (1): Let $x \in D_\sqcap$ and let $I \in \mathfrak{I}_x$. Then $x \in F \cap I$ for all $F \in \mathfrak{F}_x$; hence $I \in \mathfrak{F}'_x$. Conversely, let $I \in \mathfrak{F}'_x$. Suppose $x \notin I$. Then $I \cap D_\sqcap$ is an ideal of \underline{D}_\sqcap , by the dual of Lemma 2(1), and $x \in D_\sqcap \setminus I$. The ideal $I \cap D_\sqcap$ is contained in a prime ideal of \underline{D}_\sqcap the complement of which in D_\sqcap is a prime filter E of \underline{D}_\sqcap containing x . By Lemma 2(2), $F := \{y \in D \mid w \sqsubseteq y \text{ for some } w \in E\}$ is a filter of \underline{D} with $F \cap D_\sqcap = E$; hence $F \in \mathfrak{F}_x$. But $F \cap I = \emptyset$ which contradicts $I \in \mathfrak{F}'_x$. Thus, $x \in I$ and so $I \in \mathfrak{I}_x$. This proves $\mathfrak{F}'_x = \mathfrak{I}_x$. The corresponding equality in (2) follows dually. \square

Lemma 4 *The relation \sqsubseteq on a generalized double Boolean algebra \underline{D} defined by $x \sqsubseteq y : \iff x \sqsubseteq_\sqcap y$ and $y \sqsubseteq_\sqcup x$ ($\iff x_\sqcap = y_\sqcap$ and $x_\sqcup = y_\sqcup$) has as restriction the identity on $D_p := D_\sqcap \cup D_\sqcup$ and is a congruence relation of \underline{D} .*

Proof Let $(x, y) \in \sqsubseteq$. If $x, y \in D_\sqcap$ or $x, y \in D_\sqcup$ then, obviously, $x = y$. Now, let $x \in D_\sqcap$ and $y \in D_\sqcup$. Then $x = y_\sqcap = x_\sqcup = y_\sqcap \sqcup y_\sqcup = y_\sqcap = x_\sqcup = y$; hence $x = y$. Thus $\sqsubseteq \cap (D_p)^2 = Id_{D_p}$. Clearly, \sqsubseteq is an equivalence relation on \underline{D} which is even a congruence relation because the relationships $a_1 \sqsubseteq b_1, \dots, a_n \sqsubseteq b_n$ always imply $(a_i)_\sqcap = (b_i)_\sqcap$ and $(a_i)_\sqcup = (b_i)_\sqcup$ for $i = 1, \dots, n$ and therefore $t(a_1, \dots, a_n) = t(b_1, \dots, b_n)$ for each proper algebraic term $t(x_1, \dots, x_n)$. \square

For formulating the next two lemmas adapted from [Wi00a] too, we need the following notion: a map α between quasi-ordered sets is called *quasi-injective* if it satisfies the equivalence $x \sqsubseteq y \iff \alpha(x) \sqsubseteq \alpha(y)$.

Lemma 5 *Let \underline{D} be a generalized double Boolean algebra. Then $x \mapsto (\mathfrak{F}_x, \mathcal{I}_x)$ describes a quasi-injective homomorphism ι from \underline{D} to $\mathfrak{B}(\mathbb{K}(\underline{D}))$ having \sqsubseteq as kernel.*

Proof Let $x \in D$ and let $I \in \mathcal{I}_x$. Since $x \in F \cap I$ for all $F \in \mathfrak{F}_x$, we obtain $I \in \mathfrak{F}'_x$, i.e. $\mathcal{I}_x \subseteq \mathfrak{F}'_x$. Thus, $(\mathfrak{F}_x, \mathcal{I}_x)$ is a preconcept of $\mathbb{K}(\underline{D})$ for all $x \in D$. For $x_\sqcap \not\sqsubseteq y_\sqcap$ in \underline{D}_\sqcap there exists always an $F \in \mathfrak{F}_p(\underline{D})$ with $x_\sqcap \in F$ but $y_\sqcap \notin F$; hence $\mathfrak{F}_x = \mathfrak{F}_{x_\sqcap} \neq \mathfrak{F}_{y_\sqcap} = \mathfrak{F}_y$ and so $(\mathfrak{F}_x, \mathcal{I}_x) \neq (\mathfrak{F}_y, \mathcal{I}_y)$. Such inequality can be analogously obtained for $y_\sqcap \not\sqsubseteq x_\sqcap$, $x_\sqcup \not\sqsubseteq y_\sqcup$, and $y_\sqcup \not\sqsubseteq x_\sqcup$. If $x_\sqcap \sqsubseteq y_\sqcap$, $y_\sqcap \sqsubseteq x_\sqcap$, and $x_\sqcup \sqsubseteq y_\sqcup$, $y_\sqcup \sqsubseteq x_\sqcup$, then $x_\sqcap = y_\sqcap$ and $x_\sqcup = y_\sqcup$; hence $(\mathfrak{F}_x, \mathcal{I}_x) = (\mathfrak{F}_y, \mathcal{I}_y)$. Therefore $x \mapsto (\mathfrak{F}_x, \mathcal{I}_x)$ describes a quasi-injective map ι from \underline{D} to $\mathfrak{B}(\mathbb{K}(\underline{D}))$ having \sqsubseteq as kernel. It is even a homomorphism because, besides $\mathfrak{F}_\top = \mathfrak{F}_p(\underline{D})$ and $\mathcal{I}_\perp = \mathcal{I}_p(\underline{D})$, we can show that $\mathfrak{F}_{x \sqcap y} = \mathfrak{F}_x \cap \mathfrak{F}_y$, $\mathcal{I}_{x \sqcup y} = \mathcal{I}_x \cup \mathcal{I}_y$, $\mathfrak{F}_{\neg x} = \mathfrak{F}_p(\underline{D}) \setminus \mathfrak{F}_x$, and $\mathcal{I}_{\neg x} = \mathcal{I}_p(\underline{D}) \setminus \mathcal{I}_x$. These equalities result from Lemma 3 and the following equivalences and their duals: $F \in \mathfrak{F}_{x \sqcap y} \iff x \sqcap y \in F \iff x, y \in F \iff F \in \mathfrak{F}_x \cap \mathfrak{F}_y$ and $F \in \mathfrak{F}_{\neg x} \iff \neg x \in F \iff \neg(x_\sqcap) \in F \iff x_\sqcap \notin F \iff x \notin F \iff F \in \mathfrak{F}_p(\underline{D}) \setminus \mathfrak{F}_x$. \square

Lemma 6 *Let \underline{D} be a generalized double Boolean algebra. Then each map $\beta : \iota(D) \rightarrow D$ satisfying $\iota\beta(\mathfrak{r}) = \mathfrak{r}$ is an injective homomorphism from the image of ι into \underline{D} .*

Proof Let β be a map from $\iota(D)$ into D satisfying $\iota\beta(\mathfrak{r}) = \mathfrak{r}$. Obviously, β is injective and even bijective on $\iota(D_p)$. Since the operations of \underline{D} always result in elements of D_p , we obtain $\beta\iota(x) \sqcap \beta\iota(y) = \beta\iota(\beta\iota(x) \sqcap \beta\iota(y)) = \beta(\iota\beta\iota(x) \sqcap \iota\beta\iota(y)) = \beta(\iota(x) \sqcap \iota(y))$ and, analogously, the compatibility conditions for the other operations; hence β is an injective homomorphism. \square

Now, we are prepared to prove that each equation valid in all preconcept algebras is a logical consequence of the equations in Proposition 2. This claim is the content of the following theorem:

Theorem 2 *The equational class generated by all preconcept algebras of formal contexts is the class of all generalized double Boolean algebras.*

Proof Let \underline{D} be a generalized double Boolean algebra. For a non-trivial equation $t_1(x_1, \dots, x_n) = t_2(x_1, \dots, x_n)$ valid in all preconcept algebras, Lemma 5 yields $t_1(\iota(a_1), \dots, \iota(a_n)) = t_2(\iota(a_1), \dots, \iota(a_n))$ for $a_1, \dots, a_n \in D$ and, by Lemma 6, even $t_1(a_1, \dots, a_n) = t_1(\beta\iota(a_1), \dots, \beta\iota(a_n)) = t_2(\beta\iota(a_1), \dots, \beta\iota(a_n)) = t_2(a_1, \dots, a_n)$ for each map β from $\iota(D)$ to D with $\iota\beta(\mathfrak{x}) = \mathfrak{x}$ and $\beta\iota(a_i) = a_i$ for $i = 1, \dots, n$. With such a map β , we obtain that $a_i \sqsubseteq a_j$ implies $a_i = \beta\iota(a_i) = \beta\iota(a_j) = a_j$. In general, $a_i \sqsubseteq a_j$ ($i < j$) causes the equality $t_k(a_1, \dots, a_i, \dots, a_j, \dots, a_n) = t_k(a_1, \dots, a_i, \dots, a_i, \dots, a_n)$ for $k = 1, 2$. All together show that the equation $t_1(x_1, \dots, x_n) = t_2(x_1, \dots, x_n)$ is valid in all generalized double Boolean algebras too. \square

For a finite generalized double Boolean algebra \underline{D} , the elements of $\mathfrak{F}_p(\underline{D})$ are just the principal filters $[a] := \{x \in D \mid a \sqsubseteq x\}$ where a is an atom of the Boolean algebra \underline{D}_\sqcap , and the elements of $\mathfrak{I}_p(\underline{D})$ are just the principal ideals $[c] := \{x \in D \mid x \sqsubseteq c\}$ where c is a coatom of the Boolean algebra \underline{D}_\sqcup ; furthermore, $[a]\Delta[c] \Leftrightarrow a \sqsubseteq c$. Therefore, the formal context $(A(\underline{D}_\sqcap), C(\underline{D}_\sqcup), \sqsubseteq)$, for which $A(\underline{D}_\sqcap)$ is the set of all atoms of \underline{D}_\sqcap and $C(\underline{D}_\sqcup)$ is the set of all coatoms of \underline{D}_\sqcup , can be viewed as a simplified version of the standard context of \underline{D} . Substituting this simplified version in Lemma 5 yields the following corollary:

Corollary 1 *Let \underline{D} be a finite generalized double Boolean algebra. Then $x \mapsto (\{a \in A(\underline{D}_\sqcap) \mid a \sqsubseteq x\}, \{c \in C(\underline{D}_\sqcup) \mid x \sqsubseteq c\})$ describes a quasi-injective homomorphism ι from \underline{D} to $\mathfrak{P}(A(\underline{D}_\sqcap), C(\underline{D}_\sqcup), \sqsubseteq)$ which maps \underline{D}_p isomorphically onto $\underline{\mathfrak{H}}(A(\underline{D}_\sqcap), C(\underline{D}_\sqcup), \sqsubseteq)$.*

Acknowledgement

This paper has substantially benefited from the financial support given by the Deutsche Forschungsgemeinschaft which allowed the author to work at the Center for the Study of Language and Information (CSLI) at the Stanford University for five weeks in 2003.

References

- [Bo54] G. Boole: *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. Macmillan 1854. Reprinted by Dover Publ., New York 1958.
- [DK03] F. Dau, J. Klinger: *From Formal Concept Analysis to Contextual Logic*. FB4-Preprint, TU Darmstadt 2003.
- [DP92] B. A. Davey, H. Priestley: *Introduction to lattices and order*. Cambridge University Press, Cambridge 1990.
- [GW99a] B. Ganter, R. Wille: *Formal Concept Analysis: mathematical foundations*. Springer, Heidelberg 1999; German version: Springer, Heidelberg 1996.
- [GW99b] B. Ganter, R. Wille: Contextual Attribute Logic. In: W. Tepfenhart, W. Cyre (eds.): *Conceptual structures: standards and practices*. LNAI 1640. Springer, Heidelberg 2000, 377–388.

- [HLSW00] C. Herrmann, P. Luksch, M. Skorsky, R. Wille: Algebras of semiconcepts and double Boolean algebras. In: *Contributions to General Algebra* **13**. Verlag Johannes Heyn, Klagenfurt 2001, 175–188.
- [KV03] J. Klinger, B. Vormbrock: Contextual Boolean Logic: how did it develop? In: B. Ganter, A. de Moor (eds.): *Using conceptual structures. Contributions to ICCS 2003*. Shaker Verlag, Aachen 2003, 143–156.
- [LW91] P. Luksch, R. Wille: A mathematical model for conceptual knowledge systems. In: H. H. Bock, P. Ihm (eds.): *Classification, data analysis, and knowledge organisation*. Springer, Heidelberg 1991, 156–162.
- [Sch90] E. Schröder: *Algebra der Logik*. Bd.1. Leipzig 1890; published again by Chelsea Publ. Comp., New York 1966.
- [Se01] Th. B. Seiler: *Begreifen und Verstehen. Ein Buch über Begriffe und Bedeutungen*. Verlag Allgemeine Wissenschaft, Mühlthal 2001.
- [SW86] J. Stahl, R. Wille: Preconcepts and set representations of contexts. In: W. Gaul, M. Schader (eds.): *Classification as a tool of research*. North-Holland, Amsterdam 1986, 431–438.
- [VW03] B. Vormbrock, R. Wille: Semiconcept and protoconcept algebras: the basic theorems. FB4-Preprint, TU Darmstadt 2003.
- [Wi82] R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets*. Reidel, Dordrecht-Boston 1982, 445–470.
- [Wi92] R. Wille: Concept lattices and conceptual knowledge systems. *Computers & Mathematics with Applications* **23** (1992), 493–515.
- [Wi94] R. Wille: Plädoyer für eine philosophische Grundlegung der Begrifflichen Wissensverarbeitung. In: R. Wille, M. Zickwolff (eds.): *Begriffliche Wissensverarbeitung: Grundfragen und Aufgaben*. B.I.-Wissenschaftsverlag, Mannheim 1994, 11–25.
- [Wi96] R. Wille: Restructuring mathematical logic: an approach based on Peirce's pragmatism. In: A. Ursini, P. Agliano (eds.): *Logic and Algebra*. Marcel Dekker, New York 1996, 267–281.
- [Wi00a] R. Wille: Boolean Concept Logic. In: B. Ganter, G. W. Mineau (eds.): *Conceptual structures: logical, linguistic, and computational issues*. LNAI **1867**. Springer, Heidelberg 2000, 317–331.
- [Wi00b] R. Wille: Contextual Logic summary. In: G. Stumme (ed.): *Working with Conceptual Structures. Contributions to ICCS 2000*. Shaker, Aachen 2000, 265–276.
- [Wi01] R. Wille: Boolean Judgment Logic. In: H. Delugach, G. Stumme (eds.): *Conceptual structures: broadening the base*. LNAI **2120**. Springer, Heidelberg 2001, 115–128.
- [Wi03a] R. Wille: Conceptual content as information - basics for Conceptual Judgment Logic. In: A. de Moor, W. Lex, B. Ganter (eds.): *Conceptual structures for knowledge creation and communication*. LNAI **2746**. Springer, Heidelberg 2003, 1–15.
- [Wi03b] R. Wille: Formal Concept Analysis as mathematical theory of concepts and concept hierarchies. FB4-Preprint, TU Darmstadt 2003.

Protoconcept Graphs: The Lattice of Conceptual Contents

Joachim Hereth Correia and Julia Klinger

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D-64289 Darmstadt,
{hereth,jklinger}@mathematik.tu-darmstadt.de

Abstract. Protoconcept graphs are part of Contextual Judgment Logic. Generalizing the well-developed theory of concept graphs, they express judgments with a negation on the level of concepts and relations by representing information given in a power context family in a rhetorically structured way. The conceptual content of a protoconcept graph is understood as the information which is represented in the graph directly, enlarged by the information deducible from it by protoconcept implications of the power context family. The main result of this paper is that conceptual contents of protoconcept graphs of a given power context family can be derived as extents of the so-called conceptual information context of the power context family, thus a generalization of the Basic Theorem on $\overline{\mathbb{K}}$ -Conceptual Contents in [Wi03].

1 Introduction

The theory of protoconcept graphs is part of a program called ‘Contextual Logic’ which can be understood as a formalization of the traditional philosophical logic with its doctrine of concepts, judgments, and conclusions (cf. [Wi00b], [DK03]). Concepts, as basic units of thinking, are already formalized in Formal Concept Analysis (FCA) and its extensions ([GW99], [Wi02], [VW03]). Judgments are then understood as meaningful combinations of concepts, and with conclusions new judgments are inferred from already existing ones.

A main goal of FCA from its very beginning has been the support of rational communication. This claim has been met by using diagrammatic representations to make complex data available: Information given in a formal context is represented by a labelled line diagram, which allows even the unfamiliar user to understand and analyze the inner structure of the data. Similarly, for the doctrine of judgments, a formalization would be desirable which is both mathematically precise and representable by diagrams which are easily readable. Moreover, in order to make more complex information intelligible, it would be helpful if the diagrams could be structured rhetorically.

A promising approach is to use the system of conceptual graphs by John Sowa (see [So84],[So92]). These graphs are a diagrammatic system of logic whose purpose is ‘to express meaning in a form that is logically precise, humanly readable, and computationally tractable’ (see [So92]). The philosophical background of

this system is similar to FCA, and the system allows us to formulate judgments and conclusions in a way which is much nearer to human reasoning than predicate logic (for a more thorough discussion of the intentions and the philosophical background of FCA and Conceptual Graphs we refer to [DK03]). However, the system of conceptual graphs is not elaborated mathematically. One reason for this is that conceptual graphs were designed to be of use in a wide variety of different fields (the most prominent being knowledge representation, software specification and modelling, information extraction and natural language processing), which lead to a broad range of modifications and extensions of the core theory. The system of conceptual graphs as a whole is huge, without sharp borders and contains several ambiguities. Thus, when making conceptual graphs mathematically explicit, only restricted parts of Sowa's theory can be covered at once.

The first approach for a mathematization was discussed in [Wi97], where so-called *concept graphs* were introduced semantically. They are defined with respect to a given family of formal contexts (called *power context family*) and contain information ranging over various of these contexts. Thus, a concept graph represents information from the power context family, but it is supplied with a rhetoric structure and the information is not separated with respect to the individual contexts. In Figure 1 we see an example for a concept graph, albeit without a power context family it refers to. The power context family is omitted, because the purpose of the example is to visualize how rhetoric structure can be represented in a graph. The concept graph can be read rather straightforward: 'Tom is a cat, Jerry is a mouse and Tom chases Jerry' (or, in short, 'The cat Tom chases the mouse Jerry').

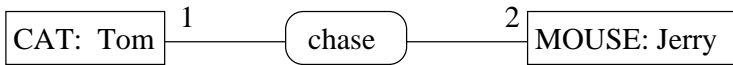


Fig. 1. A concept graph

Although it is possible to define a single graph which captures every information given in a power context family (cf. [Da03a]), usually only parts of the information are (purpose-oriented) represented in a graph. While the theory of concept graphs is well developed and includes numerous of Sowa's extensions (see [Pr98], [Wi02], [SW03]), it lacks a negation. Such a negation can be considered on different levels: In this paper we will consider so called *protoconcept graphs* which feature a (limited) negation on the level of concepts and relations; on the level of judgments negation has been introduced and discussed extensively in [Da03b]. This approach, however, leads to the classical mathematical negation and requires a separation in syntax and semantics. Since, in the present paper, we are interested in a semantic approach, Dau's graphs are not considered here.

It is necessary to discuss how the information given in a (proto)concept graph might be represented mathematically. Since concept graphs are always

defined with respect to a given power context family, the background knowledge is assumed to be codable in formal contexts. But why do formal contexts provide a suitable way of coding background knowledge without being too restrictive? One of the results of [Wi03] was that if the background knowledge consists of object- and concept implications as described by Brandom in his theory of discursive practice (cf. [Br94]), then it can be represented appropriately by formal contexts. Hence, in this paper we assume that our background knowledge is given via a power context family and refer the interested reader to [Wi03].

Now we can describe which information is transmitted by a concept graph. Obviously, the graph conveys every information unit which is expressed explicitly. For example, the sample concept graph in Figure 1 obviously contains the information that Tom is a cat. However, a (proto)concept graph may convey information which is not represented in it directly, but results from the activation of background knowledge. For instance, for the concept graph represented in Figure 1, the information that Tom is a cat may, depending on the preknowledge coded in the power context family, also transmit that Tom is an animal. The sum of all these (directly and indirectly) represented information units then constitutes the so-called *conceptual content* of the concept graph.

As argued above, the formalizations of the three doctrines are inter-related. In [Wi03], it was even shown that concept graphs as (affirmative) judgments can be made accessible through FCA. This was done by proving that the conceptual contents of concept graphs are formally derivable as concept extents. This paper aims at generalizing the result of [Wi03] to protoconcept graphs (i.e. judgments with a negation on the level of concepts and relations), thereby strengthening the inter-relationship of FCA and Boolean Judgment Logic (see [Wi01]). In particular, we will find that although protoconcept graphs are based on the more general structure of protoconcept algebras instead of concept lattices, it is nevertheless possible to express the conceptual content as extent of a suitably chosen context. Moreover, a closer study of the closure system of conceptual contents reveals a rather simple structure, which may simplify the problem of drawing these lattices.

This paper consists of three more sections. In Section 2, the basic definitions are introduced and explained by an example. In the third section, it is shown that the conceptual contents of protoconcept graphs of a given power context family (see [Wi02]) can be described as extents of a formal context. This is done in two steps: First, power context families consisting of a single context only are considered, then the result is extended to power context families of limited type in general. Section 4 contains some suggestions for further research.

2 Basic Definitions

In this section, we focus on reviewing several definitions. We assume that the reader is familiar with all basic notions of FCA (for an extensive introduction see [GW99]). First we recall several definitions from [Wi02]: Let $\mathbb{K} := (G, M, I)$ be a formal context. A *protoconcept* of \mathbb{K} is a pair (A, B) with $A \subseteq G$ and $B \subseteq M$

such that $A^I = B^{II}$ (which is equivalent to $A^{II} = B^I$). The set $\mathfrak{P}(\mathbb{K})$ of all protoconcepts of \mathbb{K} is structured by the *generalization order* \sqsubseteq , defined by

$$(A_1, B_1) \sqsubseteq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2 \text{ and } B_1 \supseteq B_2,$$

and by operations defined as follows:

$$\begin{aligned} (A_1, B_1) \sqcap (A_2, B_2) &:= (A_1 \cap A_2, (A_1 \cap A_2)^I) \\ (A_1, B_1) \sqcup (A_2, B_2) &:= ((B_1 \cap B_2)^I, B_1 \cap B_2) \\ \neg(A, B) &:= (G \setminus A, (G \setminus A)^I) \\ \neg(A, B) &:= ((M \setminus B)^I, M \setminus B) \\ \top &:= (G, \emptyset) \\ \perp &:= (\emptyset, M). \end{aligned}$$

The set $\mathfrak{P}(\mathbb{K})$ together with the operations $\sqcap, \sqcup, \neg, \neg, \top$ and \perp is called the *algebra of protoconcepts* of \mathbb{K} and denoted by $\mathfrak{P}(\mathbb{K})$. The operations are called *meet*, *join*, *negation*, *opposition*, *all* and *nothing*. Figure 2 shows an example of a formal context with its protoconcept algebra. The elements except for \top and \perp are numbered in order to make them more easily accessible for reference.

Moreover, we define a *semiconcept* of \mathbb{K} as a pair (A, B) with $A \subseteq G$ and $B \subseteq M$ such that $A^I = B$ or $B^I = A$ and define $\mathfrak{H}(\mathbb{K})$ to be the set of all semiconcepts of \mathbb{K} . Obviously, each semiconcept is a protoconcept. In particular, we define the sets $\mathfrak{H}_{\sqcap}(\mathbb{K}) := \{(A, A^I) \mid A \subseteq G\}$ and $\mathfrak{H}_{\sqcup}(\mathbb{K}) := \{(B^I, B) \mid B \subseteq M\}$ of \sqcap -semiconcepts and \sqcup -semiconcepts, respectively. The \sqcap -semiconcepts of the algebra in Figure 2 are marked by circles with the lower half filled, those which are elements of \mathfrak{H}_{\sqcup} are represented by circles whose upper half is filled. Those circles which are completely filled are concepts. The concept lattice of the context always consists of the elements in the intersection of the \sqcup - and the \sqcap -semiconcepts. Note that whenever an operation is performed, we obtain a semiconcept. In particular, the result of the operations \sqcap, \neg, \perp is a \sqcap -semiconcept, and any result of the operations \sqcup, \neg, \top is a \sqcup -semiconcept. It is a well-known fact that $\underline{\mathfrak{H}}_{\sqcap}(\mathbb{K}) := (\mathfrak{H}_{\sqcap}(\mathbb{K}), \sqcap, \sqcup, \neg, \top, \perp)$ (where $a \sqcup b := \neg(\neg a \sqcap \neg b)$ and $\top := \neg \perp$) is isomorphic to the Boolean algebra of all subsets of G . Now we define $\overline{\gamma}: G \rightarrow \mathfrak{H}_{\sqcap}(\mathbb{K})$, $\overline{\gamma}(g) = (\{g\}, \{g\}^I)$ and $\overline{\mu}: M \rightarrow \mathfrak{H}_{\sqcup}(\mathbb{K})$, $\overline{\mu}(m) = (\{m\}^I, \{m\})$. The set of \sqcup -irreducible elements of that lattice is $\{\overline{\gamma}(g) \mid g \in G\}$ and the set of \sqcap -irreducible elements is equal to $\{(G \setminus \{g\}, (G \setminus \{g\})^I) \mid g \in \mathfrak{G}\}$ (for a detailed discussion see [Wi00a]). The objects and attributes in Figure 2 are attached to the circles corresponding to their images under $\overline{\gamma}$ and $\overline{\mu}$, respectively.

Definition 1. A *power context family* $\overrightarrow{\mathbb{K}} := (\mathbb{K}_k)_{k=0,1,2,\dots}$ is a family of contexts $\mathbb{K}_k := (G_k, M_k, I_k)$ such that $G_k \subseteq G_0^k$ for $k \in \mathbb{N}$. The power context family is said to be of *limited type* $n \in \mathbb{N}$ if $\overrightarrow{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \dots, \mathbb{K}_n)$, otherwise it is called *unlimited*. The elements of $\mathfrak{P}(\mathbb{K}_0)$ are called *protoconcepts*, the elements of $\mathfrak{R}_{\mathbb{K}} := \bigcup_{k \in \mathbb{N}} \mathfrak{P}(\mathbb{K}_k)$ are called *relation protoconcepts*.

As an example for a power context family consider $\overrightarrow{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2)$ with the contexts \mathbb{K}_0 and \mathbb{K}_2 in Figure 3. The context \mathbb{K}_1 is the empty context.

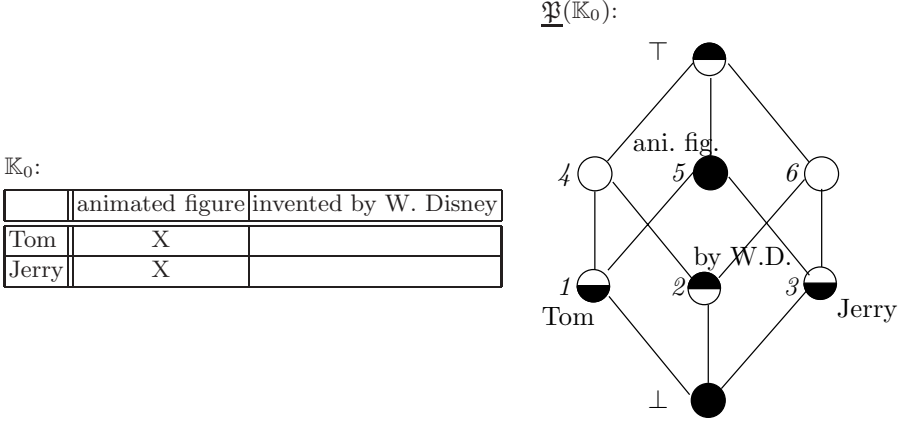


Fig. 2. A formal context and its protoconcept algebra

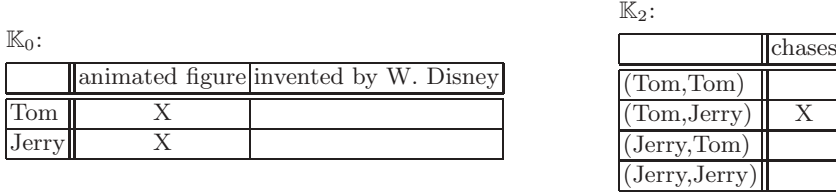


Fig. 3. A power context family

Next, we repeat the definition of protoconcept graphs as introduced in [Wi02]: The underlying structure is a so-called *relational graph*, which is a triple (V, E, ν) consisting of a set V of vertices, a set E of edges and a mapping $\nu: E \rightarrow \bigcup_{k=1,2,\dots} V^k$ which maps each edge to the ordered tuple of its adjacent vertices. If $\nu(e) = (v_1, \dots, v_k)$, then we say that the *arity* of e is k . The vertices are said to have arity 0, i.e. we set $E^{(0)} := V$. Moreover, let $E^{(k)}$ ($k = 0, 1, \dots$) be the set of all k -ary edges.

Definition 2. A *protoconcept graph* of a power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \dots)$ with $\mathbb{K}_k := (G_k, M_k, I_k)$ for $k = 0, 1, 2, \dots$ is a structure $\mathfrak{G} := (V, E, \nu, \kappa, \varrho)$ for which

- (V, E, ν) is a relational graph,
- $\kappa: V \cup E \rightarrow \bigcup_{k=0,1,\dots} \mathfrak{P}(\mathbb{K}_k)$ is a mapping $\kappa(u) \in \mathfrak{P}(\mathbb{K}_k)$ for all $u \in E^{(k)}$ ($k = 0, 1, \dots$),
- $\varrho: V \rightarrow \mathcal{P}(G_0) \setminus \{\emptyset\}$ is a mapping with $\varrho^+(v) := \varrho(v) \cap \text{Ext}(\kappa(v))$ and $\varrho^-(v) := \varrho(v) \setminus \varrho^+(v)$ satisfying that, for $\nu(e) = (v_1, \dots, v_k)$, $\varrho^+(v_j) \neq \emptyset$ for all $j = 1, \dots, k$ or $\varrho^-(v_j) \neq \emptyset$ for all $j = 1, \dots, k$, $\varrho^+(v_1) \times \dots \times \varrho^+(v_k) \subseteq \text{Ext}(\kappa(e))$ and $\varrho^-(v_1) \times \dots \times \varrho^-(v_k) \subseteq (G_0)^k \setminus \text{Ext}(\kappa(e))$.

We consider the mapping ϱ not only on vertices but also on edges: For $\nu(e) = (v_1, \dots, v_k)$, let $\varrho(e) := \varrho^+(e) \cup \varrho^-(e)$ with $\varrho^+(e) := \varrho^+(v_1) \times \dots \times \varrho^+(v_k)$ and $\varrho^-(e) := \varrho^-(v_1) \times \dots \times \varrho^-(v_k)$.

A sample protoconcept graph $\mathfrak{G} := (V, E, \nu, \kappa, \varrho)$ is shown in Figure 4. The relational graph is $(\{v, w\}, \{e\}, \nu)$ with $\nu(e) = (v, w)$; moreover we have $\kappa(v) = \overline{\gamma}(\text{Tom})$, $\kappa(w) = \overline{\gamma}(\text{Jerry})$ and $\kappa(e) = \overline{\mu}(\text{chases})$ and $\varrho(v) = \{\text{Tom}, \text{Jerry}\} = \varrho(w)$. The graph then reads: Tom chases Jerry, Jerry does not chase Tom and Tom and Jerry are animated figures (since $\overline{\gamma}(\text{Tom}) = (\{\text{Tom}\}, \{\text{animated figure}\})$ and $\overline{\gamma}(\text{Jerry}) = (\{\text{Jerry}\}, \{\text{animated figure}\})$). In particular, the graph contains the *conceptual information units* $(g, \kappa(u))$ and $(h, \neg\kappa(u))$ with $k = 0, 1, \dots$, $u \in E^{(k)}$ and $g \in \varrho^+(u)$, $h \in \varrho^-(u)$. For the graph in our example, these are the pairs $(\text{Tom}, \overline{\gamma}(\text{Tom}))$, $(\text{Jerry}, \neg\overline{\gamma}(\text{Tom}))$, $(\text{Jerry}, \overline{\gamma}(\text{Jerry}))$, $(\text{Tom}, \neg\overline{\gamma}(\text{Jerry}))$ and $((\text{Tom}, \text{Jerry}), \overline{\mu}(\text{chases}))$, $((\text{Jerry}, \text{Tom}), \neg\overline{\mu}(\text{chases}))$.

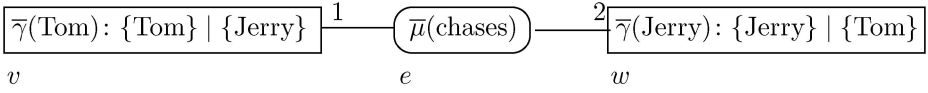


Fig. 4. A protoconcept graph of the power context family

Next, we introduce the conceptual content of a protoconcept graph in a way which is very similar to [Wi03]: First we define what protoconcept implications in \mathbb{K}_k are and then introduce the conceptual content as the disjoint union of the closures of the conceptual information units under these implications.

Thus, let $\overrightarrow{\mathbb{K}} := (\mathbb{K}_0, \dots, \mathbb{K}_n)$ be a limited power context family with $\mathbb{K}_k := (G_k, M_k, I_k)$ ($k = 0, 1, \dots, n$). For $\mathfrak{C}, \mathfrak{D} \subseteq \mathfrak{P}(\mathbb{K}_k)$, we say that the context \mathbb{K}_k satisfies $\mathfrak{C} \rightarrow \mathfrak{D}$ if $\bigcap \mathfrak{C} \subseteq \bigcap \mathfrak{D}$. In particular, we set $\mathfrak{p} \rightarrow \mathfrak{p} \sqcap \mathfrak{p}$. The formal inferences $\mathfrak{C} \rightarrow \mathfrak{D}$ give rise to a closure system $C(\mathbb{K}_k)$ on $\mathbb{S}^{imp}(\mathbb{K}_k) := \{(g, \mathfrak{p}) \in G_k \times \mathfrak{P}(\mathbb{K}_k) \mid g \in \text{Ext}(\mathfrak{p})\}$ consisting of all $Y \subseteq \mathbb{S}^{imp}(\mathbb{K})$ which satisfy

$$(P) \text{ If } A \times \mathfrak{C} \subseteq Y \text{ and if } \mathbb{K}_k \text{ satisfies } \mathfrak{C} \rightarrow \mathfrak{D} \text{ then } A \times \mathfrak{D} \subseteq Y.$$

Now the k -ary *conceptual content* $C_k(\mathfrak{G})$ for a protoconcept graph $\mathfrak{G} := (V, E, \nu, \kappa, \varrho)$ of a power context family $\overrightarrow{\mathbb{K}}$ is defined as the closure of $\{(g, \kappa(u)) \mid u \in E^{(k)} \text{ and } g \in \varrho^+(u)\} \cup \{(g, \neg\kappa(u)) \mid u \in E^{(k)} \text{ and } g \in \varrho^-(u)\}$ with respect to the closure system $C(\mathbb{K}_k)$. Then

$$C(\mathfrak{G}) := C_0(\mathfrak{G}) \dot{\cup} C_1(\mathfrak{G}) \dot{\cup} C_2(\mathfrak{G}) \dot{\cup} \dots$$

is called the $\overrightarrow{\mathbb{K}}$ -*conceptual content* of the protoconcept graph \mathfrak{G} . The 0-ary conceptual content for the graph in Figure 4 is $\mathbb{S}^{imp}(\mathbb{K}_0)$. Finally, we introduce the *information (quasi-) order* between protoconcept graphs by $\mathfrak{G}_1 \lesssim \mathfrak{G}_2 : \Leftrightarrow C(\mathfrak{G}_1) \subseteq C(\mathfrak{G}_2)$. We then say that \mathfrak{G}_1 is *less informative than* \mathfrak{G}_2 .

It is important to note that contrary to the approach in [Wi03] we do not consider so-called object implications in \mathbb{K}_k , which were defined as $A \rightarrow B : \Leftrightarrow A^{I_k} \subseteq B^{I_k}$ for $A, B \subseteq G_k$. For concepts \mathfrak{b} , $A^{I_k} \subseteq B^{I_k}$ then implies that $B \subseteq$

$\text{Ext}(\mathbf{b})$ whenever $A \subseteq \text{Ext}(\mathbf{b})$. However, in the case of protoconcept graphs we find that if two object sets satisfy $A^{I_k} \subseteq B^{I_k}$, then as soon as $A \not\subseteq B$ there are protoconcepts $\mathbf{p}_1, \mathbf{p}_2$ with $A = \text{Ext}(\mathbf{p}_1) \not\subseteq \text{Ext}(\mathbf{p}_2) = B$. In particular, for $\{g_1\}, \{g_2\}$ with $\{g_1\}^{I_k} \subseteq \{g_2\}^{I_k}$ we obtain that $\{g_1\} = \text{Ext}(\overline{\gamma}(g_1))$ and $\{g_2\} = \text{Ext}(\overline{\gamma}(g_2))$. Thus, the fact that g_2 is in the extent of a protoconcept \mathbf{p} together with $\{g_1\}^{I_k} \subseteq \{g_2\}^{I_k}$ does not imply that g_1 is in the extent of \mathbf{p} as well.

3 Conceptual Contents as Formal Concepts

Before elaborating why conceptual contents of protoconcept graphs of arbitrary power context families can be understood as extents of a formal context, we focus on the most simple case, i.e., on power context families $\vec{\mathbb{K}} := (\mathbb{K})$. Thus, *conceptual contents of a formal context* \mathbb{K} are the conceptual contents of protoconcept graphs of the power context family $\vec{\mathbb{K}} := (\mathbb{K})$ of limited type 0.

The first step is to represent the conceptual contents of \mathbb{K} in a more convenient fashion which no longer depends on the protoconcept graphs of a context, but solely on the context itself:

Definition 3. The structure $\underline{\mathbb{S}}^{imp}(\mathbb{K}) := (\mathbb{S}^{imp}(\mathbb{K}), \leq_{\mathbb{S}^{imp}(\mathbb{K})}, \widetilde{\sqcup})$ with

- (1) $(g, \mathbf{p}) \leq_{\mathbb{S}^{imp}(\mathbb{K})} (h, \mathbf{q}) :\Leftrightarrow g = h \text{ and } \mathbf{q} \sqsubseteq \mathbf{p}$
- (2) $\widetilde{\sqcup} A \times \Omega := \{(g, \bigcap_{\mathbf{p} \in \Omega} \mathbf{p}) \mid g \in A\} \text{ for } A \times \Omega \subseteq \mathbb{S}^{imp}(\mathbb{K}).$

is called the *implicational context structure* of the context \mathbb{K} . The *implicational closure* $C^{imp}(X)$ of $X \subseteq \mathbb{S}^{imp}(\mathbb{K})$ is the smallest order ideal of the structure $(\mathbb{S}^{imp}(\mathbb{K}), \leq_{\mathbb{S}^{imp}(\mathbb{K})})$ containing X closed under the partial multioperation $\widetilde{\sqcup}$. Moreover, let $C(\mathbb{S}^{imp}(\mathbb{K}))$ denote the set of all implicational closures of $\mathbb{S}^{imp}(\mathbb{K})$.

Figure 5 shows the set of all closures of $\mathbb{S}^{imp}(\mathbb{K}_0)$ ordered by set inclusion.

Lemma 4. *The implicational closures of the implicational context structure $\underline{\mathbb{S}}^{imp}(\mathbb{K})$ are exactly the conceptual contents of \mathbb{K} .*

Proof: We prove that $C(\mathbb{S}^{imp}(\mathbb{K})) = C(\mathbb{K})$: First let $U \in C(\mathbb{S}^{imp}(\mathbb{K}))$. We will prove that U satisfies the condition (P). Hence, we assume that $A \times \mathfrak{P} \subseteq U$ and that \mathbb{K} satisfy $\mathfrak{P} \rightarrow \Omega$. First we show $A \times \Omega \subseteq \mathbb{S}^{imp}(\mathbb{K})$: We have $A \subseteq \text{Ext}(\mathbf{p})$ for all $\mathbf{p} \in \mathfrak{P}$, which implies that $A \subseteq \bigcap_{\mathbf{p} \in \mathfrak{P}} \text{Ext}(\mathbf{p}) = \text{Ext}(\bigcap_{\mathbf{p} \in \mathfrak{P}} \mathbf{p})$. Since $\mathfrak{P} \rightarrow \Omega$ implies $\bigcap_{\mathbf{p} \in \mathfrak{P}} \mathbf{p} \sqsubseteq \bigcap_{\mathbf{q} \in \Omega} \mathbf{q} \sqsubseteq \mathbf{d}$ for all $\mathbf{d} \in \Omega$, we obtain $A \subseteq \text{Ext}(\mathbf{d})$ for all $\mathbf{d} \in \Omega$. Thus, $A \times \Omega \subseteq \mathbb{S}^{imp}(\mathbb{K})$. Moreover, for all $g \in A$ we have $(g, \mathbf{d}) \leq (g, \bigcap_{\mathbf{q} \in \Omega} \mathbf{q}) \leq (g, \bigcap_{\mathbf{p} \in \mathfrak{P}} \mathbf{p}) \in \widetilde{\sqcup} A \times \mathfrak{P} \subseteq U$, hence $A \times \Omega \subseteq U$. Obviously, $(U, \emptyset, \kappa, \varrho)$ with $\kappa((g, \mathbf{p})) = \mathbf{p}$ and $\varrho((g, \mathbf{p})) = \{g\}$ is a protoconcept graph of \mathbb{K} with U as conceptual content.

Now, let $U \in C(\mathbb{K})$. First we show that U is an order ideal in $\mathbb{S}^{imp}(\mathbb{K})$: Let $(g, \mathbf{p}) \in U$ and $(h, \mathbf{d}) \leq (g, \mathbf{p})$. Then $h = g$ and, since $\mathbf{p} \sqsubseteq \mathbf{d}$ implies $\mathbf{p} \sqcap \mathbf{p} \sqsubseteq \mathbf{d} \sqcap \mathbf{d}$,

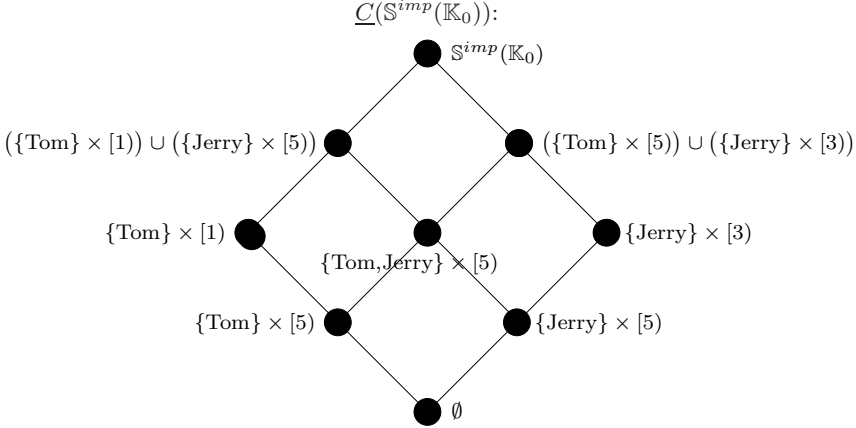


Fig. 5. The set of all closures of the implicational context structure of \mathbb{K}_0

we obtain that \mathbb{K} satisfies $\{\mathfrak{p}\} \rightarrow \{\mathfrak{d}\}$. The fact that U satisfies (P) then yields $\{g\} \times \{\mathfrak{d}\} \subseteq U$, hence $(h, \mathfrak{d}) \in U$ and U is an order ideal. Finally, let $A \times \Omega \subseteq U$. Since \mathbb{K} satisfies $\Omega \rightarrow \prod \Omega$, we obtain $\widetilde{\sqcup} A \times \Omega = A \times \{\prod \Omega\} \subseteq U$, which finishes the proof that the conceptual content U is an implicational closure of $\underline{\mathbb{S}}^{imp}(\mathbb{K})$. \square

It can easily be checked that both sets $C(\mathbb{K})$ and $C(\mathbb{S}^{imp}(\mathbb{K}))$ are closure systems. However, the set of all closures, ordered by inclusion, is always a complete lattice. We obtain that the extents of the context $(\mathbb{S}^{imp}(\mathbb{K}), C(\mathbb{S}^{imp}(\mathbb{K})), \epsilon)$ are exactly the closures, which would already yield the desired result. However, we aim at a more thorough understanding of the structure $(C(\mathbb{S}^{imp}(\mathbb{K})), \subseteq)$.

In particular, we would like to find the \wedge -irreducible elements of the complete lattice $(C(\mathbb{S}^{imp}(\mathbb{K})), \subseteq)$, thus those closures which cannot be represented as intersections of other closures. For the lattice $\underline{C}(\mathbb{S}^{imp}(\mathbb{K}))$, every element is then the intersection of a set of these \wedge -irreducible closures, resulting in both a explicitly described structure of $(C(\mathbb{S}^{imp}(\mathbb{K})), \subseteq)$ and in fewer attributes for the context whose extents will be the implicational closures. The latter follows from the fact that the concept lattice of the context with object set $\mathbb{S}^{imp}(\mathbb{K})$ and whose attributes are the \wedge -irreducible elements of $\underline{C}(\mathbb{S}^{imp}(\mathbb{K}))$ will have the closures of $\mathbb{S}^{imp}(\mathbb{K})$ as extents as well (in particular we will find that the \wedge -irreducible elements are \wedge -dense).

We will now proceed as follows: We show that $\underline{C}(\mathbb{S}^{imp}(\mathbb{K}))$ is isomorphic to a product of lattices which can be described rather easily. Then we express the \wedge -irreducible elements of the product lattice in terms of elements of the factors. Using the isomorphism, we are then able to determine the \wedge -irreducible closures in $\underline{C}(\mathbb{S}^{imp}(\mathbb{K}))$.

The next Proposition is similar to Proposition 1 in [PW99]. There, it was shown that the lattice $\underline{\Gamma}(\vec{\mathbb{K}})$ of equivalence classes of concept graphs of a given power context family $\vec{\mathbb{K}}$ is isomorphic to the subdirect product of specific sublat-

tices of the concept lattices $\mathfrak{B}(\mathbb{K}_k)$, each extended by a new \top_k -element. Here, we show that the lattice of conceptual contents of a context \mathbb{K} is isomorphic to the direct product of sublattices of the Boolean algebra $\mathfrak{H}_\sqcap(\mathbb{K})$, each extended by a new \perp -element:

Proposition 5. Let $\mathbb{K} := (G, M, I)$ be a context. We define $\mathfrak{h}_g := \mathfrak{H}_\sqcap(\mathbb{K}) \cap [\overline{\gamma}(g))$ and $V_g := (\{g\} \times \mathfrak{h}_g) \dot{\cup} \{\perp_g\}$ and define an order on V_g by $v \leq w$ if $v = \perp_g$ or $v \leq_{\mathfrak{S}^{imp}(\mathbb{K})} w$ ($v, w \in V_g$). Then $\underline{C}(\mathfrak{S}^{imp}(\mathbb{K})) \cong \prod_{g \in G} V_g$.

Proof: We prove that the mapping

$$\begin{aligned} \varphi: \prod_{g \in G} V_g &\longrightarrow \underline{C}(\mathfrak{S}^{imp}(\mathbb{K})) \\ (a_g)_{g \in G} &\longmapsto \bigcup_{\substack{g \in G, \\ a_g = (g, \mathfrak{q}_g) \in V_g \setminus \{\perp_g\}}} \{g\} \times [\mathfrak{q}_g) \end{aligned}$$

is an order-isomorphism.

First we show that φ is well defined, i.e. that $\varphi((a_g)_{g \in G}) \in C(\mathfrak{S}^{imp}(\mathbb{K}))$ for all $(a_g)_{g \in G}$. Let $X := \varphi((a_g)_{g \in G}) = \bigcup_{a_g = (g, \mathfrak{q}_g) \in V_g \setminus \{\perp_g\}} \{g\} \times [\mathfrak{q}_g)$. We need to check that X is an order ideal in $\mathfrak{S}^{imp}(\mathbb{K})$ which is closed under $\widetilde{\sqcap}$. First, let $(g, \mathfrak{p}) \in X$ and $(h, \mathfrak{q}) \leq_{\mathfrak{S}^{imp}(\mathbb{K})} (g, \mathfrak{p})$. Then $h = g$ and $\mathfrak{p} \sqsubseteq \mathfrak{q}$. However, since $(g, \mathfrak{p}) \in X$, we have $a_g \neq \perp_g$. Thus, there exists a $\mathfrak{p}_g \in \mathfrak{h}_g$ with $a_g = (g, \mathfrak{p}_g)$. Moreover, $(g, \mathfrak{p}) \in X$ implies $\mathfrak{p}_g \sqsubseteq \mathfrak{p}$. Since $\mathfrak{p} \sqsubseteq \mathfrak{q}$, this in turn implies $\mathfrak{p}_g \sqsubseteq \mathfrak{q}$. Hence $\mathfrak{q} \in [\mathfrak{p}_g)$, and therefore $(g, \mathfrak{q}) = (h, \mathfrak{q}) \in X$. Next, let $A \times \Omega \subseteq X$. Hence, for each $g \in A$ we have $\{g\} \times \Omega \subseteq X$. Then $a_g = (g, \mathfrak{q}_g)$ for some $\mathfrak{q} \in \mathfrak{h}_g$, yielding $\mathfrak{p}_g \sqsubseteq \mathfrak{q}$ for all $\mathfrak{q} \in \Omega$. Since \mathfrak{p}_g is a \sqcap -semiconcept, it is the smallest protoconcept with extent $\text{Ext}(\mathfrak{p}_g)$ and we obtain $\mathfrak{p}_g \sqsubseteq \sqcap \Omega$. Thus, for all $g \in A$ we have $(g, \sqcap \Omega) \in X$, yielding in turn that $\widetilde{\sqcap} A \times \Omega \subseteq X$. This finishes the proof that $X \in C(\mathfrak{S}^{imp}(\mathbb{K}))$.

Next we show that φ is an order-embedding, i.e. that $(a_g)_{g \in G} \leq_{V_g} (b_g)_{g \in G} \Leftrightarrow \varphi((a_g)_{g \in G}) \subseteq \varphi((b_g)_{g \in G})$. Thus, let $(a_g)_{g \in G} \leq_{\prod V_g} (b_g)_{g \in G}$. This is equivalent to $a_g \leq_{V_g} b_g$ for all $g \in G$. First we notice that $a_g = \perp_g$ is equivalent to $\varphi((a_g)_{g \in G}) \cap (\{g\} \times [\overline{\gamma}(g))) = \emptyset$. Moreover, for $a_g = (g, \mathfrak{p}_g)$ and $b_g = (g, \mathfrak{q}_g)$ we find $a_g \leq_{V_g} b_g \Leftrightarrow (g, \mathfrak{p}_g) \leq_{\mathfrak{S}^{imp}(\mathbb{K})} (g, \mathfrak{q}_g) \Leftrightarrow \mathfrak{q}_g \sqsubseteq \mathfrak{p}_g \Leftrightarrow [\mathfrak{p}_g) \subseteq [\mathfrak{q}_g) \Leftrightarrow \{g\} \times [\mathfrak{p}_g) \subseteq \{g\} \times [\mathfrak{q}_g) \Leftrightarrow \varphi((a_g)_{g \in G}) \cap (\{g\} \times [\overline{\gamma}(g))) \subseteq \varphi((b_g)_{g \in G}) \cap (\{g\} \times [\overline{\gamma}(g)))$. Hence, $(a_g)_{g \in G} \leq_{\prod V_g} (b_g)_{g \in G}$ is equivalent to the fact that for all $g \in G$ we have $\varphi((a_g)_{g \in G}) \cap (\{g\} \times [\overline{\gamma}(g))) \subseteq \varphi((b_g)_{g \in G}) \cap (\{g\} \times [\overline{\gamma}(g)))$, which in turn is equivalent to $\varphi((a_g)_{g \in G}) \subseteq \varphi((b_g)_{g \in G})$.

The fact that φ is an order-embedding implies that φ is injective. What is left to show is that φ is surjective. In order to prove this, for each $S \in C(\mathfrak{S}^{imp}(\mathbb{K}))$ we need to find an $(a_g)_{g \in G}$ with $\varphi((a_g)_{g \in G}) = S$. Thus, let $S \in C(\mathfrak{S}^{imp}(\mathbb{K}))$, let $A := \{g \in G \mid \exists \mathfrak{p} \in \mathfrak{P}(\mathbb{K}). (g, \mathfrak{p}) \in S\}$, and for each $g \in A$ we consider the set $\mathfrak{P}_g := \{\mathfrak{p} \in \mathfrak{P}(\mathbb{K}) \mid (g, \mathfrak{p}) \in S\}$. Since S is a closure, we obtain for each $g \in G$ that $\widetilde{\sqcap} \{g\} \times \mathfrak{P}_g \subseteq S$, thus $(g, \sqcap \mathfrak{P}_g) \in S$. Since $\sqcap \mathfrak{P}_g$ is a \sqcap -semiconcept by definition, we have $\sqcap \mathfrak{P}_g \in \mathfrak{h}_g$ and hence $\mathfrak{P}_g = [\sqcap \mathfrak{P}_g)$. Therefore, $S = \bigcup_{g \in A} \{g\} \times [\sqcap \mathfrak{P}_g)$.

Finally, we set $(a_g)_{g \in G}$ with

$$a_g = \begin{cases} (g, \sqcap \mathfrak{P}_g) & \text{if } g \in A \\ \perp_g & \text{if } g \notin A. \end{cases}$$

It can now be easily checked that $\varphi((a_g)_{g \in G}) = S$, finishing the proof that φ is an order-isomorphism. \square

In Figure 6 we find a visualization of Proposition 5. for $C(\mathcal{S}^{imp}(\mathbb{K}_0))$.

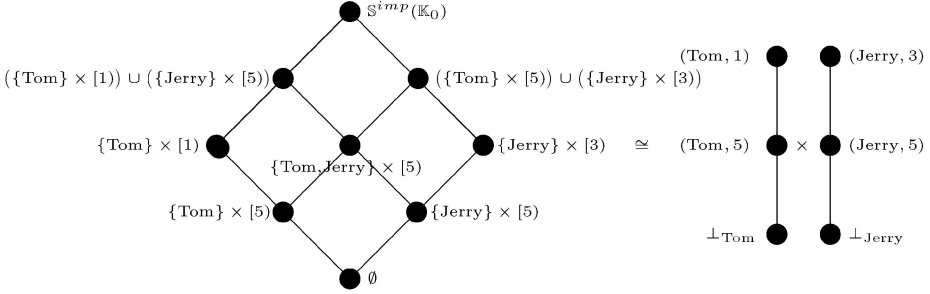


Fig. 6. Example for Proposition 5.

Since each of the lattices \underline{V}_g is complete, the product $\prod_{g \in G} \underline{V}_g$ is a complete lattice as well. The lattice \underline{V}_g corresponds to the Boolean lattice $\underline{h}_g := (\mathfrak{H}_{\sqcap}(\mathbb{K}) \cap [\overline{\gamma}(g), \sqsupset])$, with an additional \perp -element. The \wedge -irreducible elements of \underline{V}_g are therefore fairly easy to characterize: They consist of the set of coatoms $\{(\{g, h\}, (\{g, h\})^I) \in \mathfrak{h}_g \mid h \neq g\}$ of $(\mathfrak{H}_{\sqcap}(\mathbb{K}) \cap [\overline{\gamma}(g), \sqsupset])$ plus the additional element \perp_g . It is easy to check that these \wedge -irreducible elements are \wedge -dense in \underline{V}_g .

Having characterized the \wedge -irreducible elements of the factor-lattices, we may now determine the corresponding elements of the product:

Lemma 6. *Let $|G| > 1$. The element $(a_g)_{g \in G} \in \prod_{g \in G} \underline{V}_g$ is \wedge -irreducible in $\prod_{g \in G} \underline{V}_g$ if and only if there is exactly one $h \in G$ such that a_h is \wedge -irreducible in \underline{V}_h and if $a_g = \top_{\underline{V}_g}$ for all $g \neq h$. Moreover, the \wedge -irreducible elements are \wedge -dense in $\prod_{g \in G} \underline{V}_g$.*

Proof: First we show that each $(a_g)_{g \in G} \in \prod_{g \in G} \underline{V}_g$ satisfying the condition is \wedge -irreducible. Thus, let there be exactly one $h \in G$ such that a_h is \wedge -irreducible in \underline{V}_h and $a_g = \top_{\underline{V}_g}$ for all $g \neq h$. Now we assume that

$$(a_g)_{g \in G} = \bigwedge_{(b_g)_{g \in G} > (a_g)_{g \in G}} (b_g)_{g \in G}.$$

Then $b_g = \top_{\underline{V}_g} = a_g$ for all $g \neq h$ and $a_h = \bigwedge_{b_h > a_h} b_h$. This, however, contradicts the assumption that a_h is \wedge -irreducible in \underline{V}_h . Hence, $(a_g)_{g \in G}$ is \wedge -irreducible in $\prod_{g \in G} \underline{V}_g$.

Next we prove that each \wedge –irreducible element of the product-lattice has the form described above: If for $(a_g)_{g \in G} \in \prod_{g \in G} \underline{V}_g$ there are two distinct elements $h_1, h_2 \in G$ with $a_{h_1} \neq \top_{V_{h_1}}$ and $a_{h_2} \neq \top_{V_{h_2}}$, then $(a_g)_{g \in G} = (b_g)_{g \in G} \wedge (c_g)_{g \in G}$ with

$$b_g = \begin{cases} a_g & \text{if } g \neq h_1 \\ \top_{V_{h_1}} & \text{otherwise} \end{cases} \quad \text{and} \quad c_g = \begin{cases} a_g & \text{if } g \neq h_2 \\ \top_{V_{h_2}} & \text{otherwise} \end{cases}$$

Hence, $(a_g)_{g \in G}$ is not \wedge –irreducible in $\prod_{g \in G} \underline{V}_g$.

Finally, the fact that the \wedge –irreducible elements of $\prod_{g \in G} \underline{V}_g$ are \wedge –dense in the product immediately follows from their construction. \square

As an immediate result from Proposition 5. and Lemma 6. we obtain

Corollary 7. For $k \in G$, let $I(k) := \{\mathfrak{p} \in \mathfrak{P}(\mathbb{K}) \mid \mathfrak{p} \sqsubset (G \setminus \{k\}, G \setminus \{k\}^I)\}$. Then the \wedge –irreducible elements of $\underline{C}(\mathbb{S}^{imp}(\mathbb{K}))$ are exactly the elements of $M^{irr}(\mathbb{K}) := M_{\perp}^{irr}(\mathbb{K}) \cup M_{co}^{irr}(\mathbb{K})$ with

$$\begin{aligned} M_{\perp}^{irr}(\mathbb{K}) &:= \{\mathbb{S}^{imp}(\mathbb{K}) \setminus (\{g\} \times \mathfrak{P}(\mathbb{K})) \mid g \in G\} \text{ and} \\ M_{co}^{irr}(\mathbb{K}) &:= \{\mathbb{S}^{imp}(\mathbb{K}) \setminus \{g\} \times I(k) \mid g, k \in G, g \neq k\}. \end{aligned}$$

Proof: We apply the isomorphism from Proposition 5. to the \wedge –irreducible elements of Lemma 6.. \square

We may thus define in accordance to [Wi03] the so-called *conceptual information context* of \mathbb{K} as $\mathbb{K}^{inf} := (\mathbb{S}^{imp}, M^{irr}(\mathbb{K}), \in)$, enabling us to state the first main result of this paper:

Theorem 8. (Basic Theorem on \mathbb{K} –Conceptual Contents) For a formal context \mathbb{K} , the extents of the conceptual information context $\mathbb{K}^{inf}(\mathbb{K})$ are exactly the implicational closures of the implicational context structure $\mathbb{S}^{imp}(\mathbb{K})$; those implicational closures are exactly the conceptual contents of \mathbb{K} .

Proof: Since the extents of $\mathbb{K}^{inf}(\mathbb{K})$ are exactly the intersections of attribute concepts and since the elements of $C(\mathbb{S}^{imp}(\mathbb{K}))$ are exactly the intersections of $M^{irr}(\mathbb{K})$, Corollary 7. yields the first result. With Lemma 4. we then obtain the second claim. \square

We may now extend the Basic Theorem on \mathbb{K} –Conceptual Contents to the general case of (limited) power context families. The construction and the proof of the Theorem are essentially the same as in [Wi03]. However, in order to make this paper self-contained we repeat both the theorem and the proof, with a few adjustments to suit our purpose. Let $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \dots, \mathbb{K}_n)$ be a power context family with $\mathbb{K}_k := (G_k, M_k, I_k)$ ($k = 0, 1, \dots, n$). The *conceptual information context* corresponding to $\vec{\mathbb{K}}$ is defined as the formal context

$$\mathbb{K}^{inf}(\vec{\mathbb{K}}) := \mathbb{K}^{inf}(\mathbb{K}_0) + \mathbb{K}^{inf}(\mathbb{K}_1) + \dots + \mathbb{K}^{inf}(\mathbb{K}_n),$$

thus as the direct sum of the contexts $\mathbb{K}^{inf}(\mathbb{K}_k)$ ($k = 1, \dots, n$). An extent U of $\mathbb{K}^{inf}(\vec{\mathbb{K}})$ is said to be *rooted* if $((g_1, \dots, g_k), \mathfrak{b}_k) \in U$ implies $(g_j, \top_0) \in U$ for

$j = 1, \dots, k$ and $\top_0 := (G_0, G_0^{I_0})$. Rooted extents are needed in order to identify the graphs with extents of the context $\mathbb{K}^{inf}(\vec{\mathbb{K}})$ (an extent which is not rooted would correspond to a graph which has an edge but is missing at least one of the adjacent vertices). Now we are able to formulate the desired theorem:

Theorem 9. (Basic Theorem on $\vec{\mathbb{K}}$ -Conceptual Contents) For a power context family $\vec{\mathbb{K}}$ of limited type n the conceptual contents of the protoconcept graphs of $\vec{\mathbb{K}}$ are exactly the rooted extents of the corresponding conceptual information context $\mathbb{K}^{inf}(\vec{\mathbb{K}})$.

Proof: By definition, the conceptual content of a protoconcept graph is the disjoint union $C(\mathfrak{G}) := C_0(\mathfrak{G}) \dot{\cup} C_1(\mathfrak{G}) \dot{\cup} \dots \dot{\cup} C_n(\mathfrak{G})$. By Theorem 8., for each $k = 0, 1, \dots, n$ the conceptual content $C_k(\mathfrak{G})$ is an extent of $\mathbb{K}^{inf}(\mathbb{K}_k)$. Since $\mathbb{K}^{inf}(\vec{\mathbb{K}})$ is the direct sum of all these contexts, we find that $C(\mathfrak{G})$ is an extent of $\mathbb{K}^{inf}(\vec{\mathbb{K}})$. This extent is rooted as a direct consequence of the definition of protoconcept graphs. Conversely, let U be a rooted extent of $\mathbb{K}^{inf}(\vec{\mathbb{K}})$. Then $U_k := U \cap G_k$ is an extent of $\mathbb{K}^{inf}(\mathbb{K}_k)$ for each $k = 0, 1, \dots, n$ and therefore an implicational closure of $\mathbb{S}^{imp}(\mathbb{K}_k)$ by Theorem 8.. Now we define a protoconcept graph $\mathfrak{G} := (V, E, \nu, \kappa, \varrho)$ by $V := U_0$, $E := \bigcup_{k=1, \dots, n} U_k$, $\nu((g_1, \dots, g_k), \mathfrak{p}_k) := ((g_1, \top_0), \dots, (g_k, \top_0))$, $\kappa((g, \mathfrak{p}_0)) := \mathfrak{p}_0$, $\kappa(((g_1, \dots, g_k), \mathfrak{p}_k)) := \mathfrak{p}_k$, $\varrho((g, \mathfrak{p}_0)) := \{g\}$. Obviously, this protoconcept graph has U as conceptual content. \square

This result can be considered as a further step in elaborating the inter-relationships of Contextual Logic. Generalizing the result in [Wi03], it makes protoconcept graphs accessible via Formal Concept Analysis by proving that even conceptual contents of judgments with a (local) negation are derivable as formal concepts. Moreover, Proposition 5. and the two basic theorems give us some structural insight regarding the closure systems $C(\mathbb{K})$ and $C(\vec{\mathbb{K}})$ which might prove valuable for the development of TOSCANA-systems for protoconcept graphs of power context families (see [EGSW00]).

4 Outlook

In this paper, object implications in power context families were not considered at all (due to reasons which were explained at the end of Section 2). However, it seems possible to include a variant of object implications in this theory by defining a closure closed under object implications using only instances which are in $\mathbb{S}^{imp}(\mathbb{K})$. Since this approach exceeds the range of this paper, it would be desirable to continue the study of conceptual contents of protoconcept graphs.

In [Da04], it was described how a logic approach to concept graphs can effectively deal with object and concept implications via new derivation rules. It might be interesting to see if and how this theory can be transferred to a theory of protoconcept graphs which is based on a separation in syntax and semantics. In particular, the relation of such a syntactic approach and the semantic theory described in the present paper might lead to new insights with respect to the overall theory of protoconcept graphs.

References

- [Br94] R.B. Brandom: Making it explicit. Reasoning, Representing, and Discursive Commitment. Harvard University Press, Cambridge 1994.
- [Da03a] F. Dau: Concept Graphs without Negations: Standard models and Standard graphs. In: A. de Moor, W. Lex, B. Ganter (Eds.): Conceptual Structures for Knowledge Creation and Communication. Springer Verlag, Berlin–New York 2003, 243-256.
- [Da03b] F. Dau: The Logic System of Concept Graphs with Negations (and its Relationship to Predicate Logic), Lecture notes on Artificial Intelligence, Vol. 2892, Springer Verlag, Berlin – New York, 2003.
- [Da04] F. Dau: Background Knowledge in Concept Graphs. This volume.
- [DK03] F. Dau, J. Klinger: From Formal Concept Analysis to Contextual Logic. FB4-Preprint, TU Darmstadt 2003.
- [EGSW00] P. Eklund, B. Groh, G. Stumme, R. Wille: A Contextual-Logic Extension of TOSCAN. In: B. Ganter, G. W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues, Springer Verlag, Berlin–New York 2000, 453-467.
- [GW99] B. Ganter, R. Wille: Formal Concept Analysis: Mathematical Foundations. Springer Verlag, Berlin–New York 1999.
- [Pr98] S. Prediger: Kontextuelle Urteilslogik mit Begriffsgraphen, Shaker Verlag, Aachen 1998.
- [PW99] S. Prediger, R. Wille: The Lattice of Concept Graphs of a Relationally Scaled Context. In: W. Tepfenhart, W. Cyre (Eds.): Conceptual Structures: Standards and Practices, Springer Verlag, Berlin – New York 1999, 401-414.
- [SW03] L. Schoolmann, R. Wille: Concept Graphs with Subdivision: a Semantic Approach. In: U. Priss, D. Corbett, G. Angelova (Eds.): Conceptual Structures: Integration and Interfaces, Springer Verlag, Berlin–New York 2002, 271-281.
- [So84] J. F. Sowa: Conceptual Structures: Information Processing in Mind and Machine. Adison-Wesley, Reading 1984.
- [So92] J. F. Sowa: Conceptual Graphs Summary. In: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (Eds.): Conceptual Structures: Current Research and Practice. Ellis Horwood, 1992, 3-51.
- [VW03] B. Vormbrock, R. Wille: Semiconcept and Protoconcept Algebras: The Basic Theorems. FB4-Preprint, TU Darmstadt, 2003.
- [Wi97] R. Wille: Conceptual Graphs and Formal Concept Analysis. In: D. Lukose, H. Delugach, M. Keeler, L. Searle, J. Sowa (Eds.): Conceptual Structures: Fulfilling Peirce's Dream. Springer, Berlin - Heidelberg - New York 1997, 290 - 303.
- [Wi00a] R. Wille: Boolean Concept Logic. In: B. Ganter, G.W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues, Springer Verlag, Berlin–New York 2000, 317-331.
- [Wi00b] R. Wille: Contextual Logic Summary. In: G. Stumme (Ed.): Working with Conceptual Structures. Contributions to ICCS 2000. Shaker, Aachen 2000, 256-276.
- [Wi01] R. Wille: Boolean Judgment Logic. In: H. S. Delugach, G. Stumme (Eds.): Conceptual Structures: Broadening the Base, Springer Verlag, Berlin–New York 2001, 115-128.

- [Wi02] R. Wille: Existential Graphs of Power Context Families. In: U. Priss, D. Corbett, G. Angelova (Eds.): *Conceptual Structures: Integration and Interfaces*, Springer Verlag, Berlin–New York 2002, 382-396.
- [Wi03] R. Wille: Conceptual Content as Information - Basics for Contextual Judgment Logic. In: A. de Moor, W. Lex, B. Ganter (Eds.): *Conceptual Structures for Knowledge Creation and Communication*. Springer Verlag, Berlin–New York 2003, 1-15.

Signs and Formal Concepts

Uta Priss

School of Computing, Napier University
u.priss@napier.ac.uk

1 Introduction

In this paper we propose a semiotic conceptual framework which is compatible with Peirce's definition of signs and uses formal concept analysis for its conceptual structures. The goal of our research is to improve the use of formal languages such as ontology languages and programming languages. Even though there exist a myriad of theories, models and implementations of formal languages, in practice it is often not clear which strategies to use. AI ontology language research is in danger of repeating mistakes that have already been studied in other disciplines (such as linguistics and library science) years ago.

Just to give an example of existing inefficiencies: Prechelt (2000) compares the implementations of the same program in different programming languages. In an experiment he asked programmers of different languages to write a program for a certain problem. All programmers of so-called scripting languages (Perl, Python, Tcl) used associative arrays as the main data structure for their solution, which resulted in very efficient code. C++ and Java programmers did not use associative arrays but instead manually designed suitable data structures, which in many cases were not very efficient. In scripting languages associative arrays are very commonly used and every student of such languages is usually taught how to use them. In Java and C++, associative arrays are available via the class hierarchy, but not many programmers know about them. Therefore scripting languages performed better in the experiment simply because programmers of Java and C++ were not able to find available, efficient data structures within the large class libraries of these languages. Of course, this does not imply that scripting languages always perform better, but in some cases apparently large class libraries can be a hindrance.

These kinds of problems indicate that the challenges of computing nowadays lie frequently in the area of information management. A semiotic-conceptual framework as proposed in this paper views formal languages within a system of information management tasks. More specifically, it identifies management tasks relating to names (namespaces), contexts and (object) identifiers as the three contributing factors. These three management tasks correspond to the three components of a sign: representation, context and denotation.

It has been shown in the area of software engineering that formal concept analysis can be used for such information management tasks (Snelting, 1995). Snelting uses formal concept analysis for managing the dependencies of variables within legacy code. But we argue that it is not obvious how to connect the three different areas of management tasks to each other if considering only conceptual structures, because sign use involves semiotic aspects in addition to conceptual structures. The semiotic conceptual framework

described in this paper facilitates a formal description of semiotic aspects of formal languages. It predicts the roles which conceptual and semiotic aspects play in formal languages. This is illustrated in a few examples in section 8 of this paper. It should be pointed out, however, that this research is still in its beginnings. We have not yet explored the full potential of applications of this semiotic conceptual framework.

2 The Difference between Signs and Mathematical Entities

A semiotic conceptual framework contrasts signs with mathematical entities. The variables in formal logic and mathematics are mathematical entities because they are fully described by rules, axioms and grammars. Programmers might think of mathematical entities as “strings”, which have no other meaning apart from their functioning as place holders. Variables in declarative programming languages are richer entities than strings because they have a name, a data type and a value (or state) which depends on the time and context of the program when it is executed. These variables are modelled as signs in a semiotic conceptual framework.

One difference between mathematical entities and signs is the relevance of context. Mathematics employs global contexts. From a mathematical view, formal contexts in formal concept analysis are just mathematical entities. The socio-pragmatic context of an application of formal concept analysis involves signs but extends far beyond formal structures. On the other hand, computer programs are completely formal but their contexts always have a real-time spatial-temporal component, including the version of the underlying operating system and the programmer’s intentions. Computer programs cannot exist without user judgements, whereas mathematical entities are fully defined independently of a specific user.

Many areas of computing require an explicit handling of contextual aspects of signs. For example, contextual aspects of databases include transaction logs, recovery routines, performance tuning, and user support. Programmers often classify these as “error” or “exception” handling procedures because they appear to distract from the elegant, logical and deterministic aspects of computer programs. But if one considers elements of computers as “signs”, which exist in real world contexts, then maybe contextual aspects can be considered the norm whereas the existence of logical, deterministic, algorithmic aspects is a minor (although very convenient) factor.

3 A Semiotic Conceptual Definition of Signs

Peirce (1897) defines a sign as follows: “A sign, or representamen, is something which stands to somebody for something in some respect or capacity. It addresses somebody, that is, creates in the mind of that person an equivalent sign, or perhaps a more developed sign. That sign which it creates I call the interpretant of the first sign. The sign stands for something, its object.” Our semiotic conceptual framework is based on a formalisation of this definition, which is described below. To avoid confusion with the modern meaning of “object” in programming languages, “denotation” is used instead of “object”.

A representamen is a physical form for communication purposes. Representamens of formal languages, for example variable names, are considered mathematical entities

in this paper. All allowable operations among representamens are fully described by the representamen rules of a formal language. Two representamens are equal if their equality can be mathematically concluded from the representamen rules. For example, representamen rules could state that a string “4+1” is different from a string “5”, whereas for numbers: $4 + 1 = 5$.

In a semiotic conceptual framework, Peirce’s sign definition is formalised as follows: A sign is a triadic relation ($rmn(s)$, $den(s)$, $ipt(s)$) consisting of a representamen $rmn(s)$, a denotation $den(s)$ and an interpretant $ipt(s)$ where $den(s)$ and $ipt(s)$ are signs themselves (cf. figure 1). It is sometimes difficult to distinguish between a sign and its representamen, but $rmn(s)$ is used for the mathematical entity that refers to the sign and s for the sign itself.

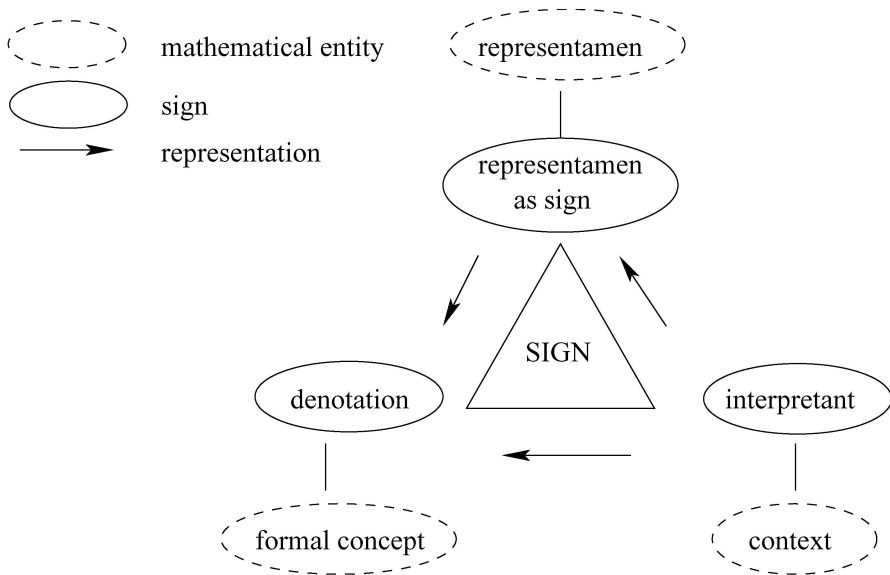


Fig. 1. The sign triad

Even though the three components of the sign are written as mappings, $rmn()$, $den()$, $ipt()$, these mappings only need to be defined with respect to the smallest possible interpretant which is the sign’s own interpretant at the moment when the sign is actually used. Further conditions for compatibility among interpretants must be provided (see below) before triadic sign relations can be considered across several or larger interpretants. Such compatibility conditions must exist because otherwise signs would be completely isolated from each other and communication would not be possible.

Interpretants and denotations are signs themselves with respect to other interpretants. But they also relate to mathematical entities. Denotations relate to (formal) concepts. Interpretants relate to contexts, which are in this paper defined as the formalisable aspects of interpretants. Interpretants contain more information than contexts. According to

Peirce, interpretants mediate between representamens and denotations. Therefore an interpretant or context only needs to contain as much information as is needed for understanding a sign. Because “a sign, or representamen, is something which stands to somebody for something in some respect or capacity” (Peirce, 1897), it follows that signs are not hypothetical but actually exist in the present or have existed in the past.

4 Synonymy and Similar Sign Equivalences

The definition of signs in a semiotic conceptual framework does not guarantee that representamens are unambiguous and represent exactly one denotation with respect to one interpretant. Furthermore, the definition does not specify under which conditions a sign is equal to another sign or even to itself. Conditions for interpretants must be described which facilitate disambiguation and equality.

A set I of n interpretants, i_1, i_2, \dots, i_n , is called overlapping iff

$$\forall a, 1 \leq a \leq n \exists b, 1 \leq b \leq n, b \neq a \exists s : i_a \rightarrow s, i_b \rightarrow s \quad (1)$$

where s denotes a sign. The arrow relation “ \rightarrow ” is called “representation” and is the same as in figure 1. This relation is central to Peirce’s definition of signs but shall not be further discussed in this paper.

With respect to a set I of overlapping interpretants, any equivalence relation can be called synonymy, denoted by \equiv_I , if the following necessary condition is fulfilled for all signs represented by interpretants in I :

$$(rmn(s_1), den(s_1), ipt(s_1)) \equiv_I (rmn(s_2), den(s_2), ipt(s_2)) \implies s_1 \rightarrow den(s_2), s_2 \rightarrow den(s_1), den(s_1) \equiv_I den(s_2) \quad (2)$$

Only a necessary but not sufficient condition is provided for synonymy because it depends on user judgements. In a programming language, synonymy can be a form of assignment. If a programmer assigns a variable to be a pointer (or reference) to another variable’s value, then these two variables are synonyms. Denotational equality is usually not required for synonymous variables because values can change over time. For example two variables, “counter := 5” and “age := 5”, are not synonymous just because they have the same value. Because synonymy is an equivalence relation, signs are synonymous to themselves.

A further condition for interpretants ensures disambiguation of representamens: A set I of overlapping interpretants is called compatible iff

$$\forall i_1, i_2 \in I \forall s_1, s_2 : (i_1 \rightarrow s_1, i_2 \rightarrow s_2, rmn(s_1) = rmn(s_2) \implies s_1 \equiv_I s_2) \quad (3)$$

In the rest of this paper, all single interpretants are always assumed to be compatible with themselves. Compatibility between interpretants can always be achieved by renaming of signs, for example, by adding a prefix or suffix to a sign.

The following other equivalences are defined for signs in a set I of compatible interpretants:

$$\text{identity: } s_1 \asymp_I s_2 :\iff id(s_1) = id(s_2), \quad s_1 \equiv_I s_2 \quad (4)$$

$$\text{polysemy: } s_1 \dot{=} s_2 :\iff rmn(s_1) = rmn(s_2) \quad (5)$$

$$\text{equality: } s_1 =_I s_2 :\iff den(s_1) =_I den(s_2), rmn(s_1) = rmn(s_2) \quad (6)$$

$$\text{equinymy: } s_1 \cong_I s_2 \implies den(s_1) =_I den(s_2), s_1 \equiv_I s_2 \quad (7)$$

$$s_1 \cong_I s_2 \iff s_1 =_I s_2$$

Identity refers to what is called “object identifiers” in object-oriented languages whereas equinymy is a form of value equality. For example, in a program sequence, “age := 5, counter := 5, age := 6”, the variables “age” and “counter” are initially equal. But “age” is identical to itself even though it changes its value. Identity is implemented via a set \mathcal{I} of mathematical entities. The elements of \mathcal{I} are called identifiers. A mapping $id()$ maps a sign onto an identifier or onto NULL if the sign does not have an identifier. It should be required that if two signs are equal and one of them has an identifier then both signs are also identical. The only operation or relation that is available for identifiers is “=”. In contrast to synonymy which is asserted by users, object-oriented programming languages and databases have rules for when and how to create “object identifiers”.

Because the relations in (4)-(7) are equivalence relations, signs are identical, polysemous, equinymous and equal to themselves. In (5) polysemy is defined with respect to equal representamens but only in compatible interpretants. Signs with equal representamens across non-compatible interpretants are often called “homographs”. This definition of “polysemy” is different from the one in linguistics which does not usually imply synonymy.

The following statements summarise the implications among the relations in (4)-(7).

$$s_1 \asymp s_2 \text{ or } s_1 \dot{=} s_2 \text{ or } s_1 \cong_I s_2 \implies s_1 \equiv_I s_2 \quad (8)$$

$$s_1 = s_2 \iff s_1 \dot{=} s_2, \quad s_1 \cong_I s_2 \quad (9)$$

5 Anonymous Signs and Mergeable Interpretants

Two special cases are of interest: anonymous signs and mergeable interpretants. In programming languages, anonymous signs are constants. An anonymous sign with respect to compatible interpretants I is defined as a sign with

$$s =_I den(s) \quad (10)$$

An anonymous sign denotes itself and has no other representamen than the representamen of its denotation. Signs which are anonymous with respect to one interpretant need not be anonymous with respect to other interpretants.

The following equations and statements are true for anonymous signs s, s_1, s_2

$$s =_I den(s) \implies s =_I den(s) =_I den(den(s)) =_I \dots \quad (11)$$

$$s =_I den(s) \implies rmn(s) = rmn(den(s)) = rmn(den(den(s))) = \dots \quad (12)$$

$$s_1 =_I s_2 \iff den(s_1) =_I den(s_2) \quad (13)$$

$$s_1 =_I s_2 \iff s_1 \cong_I s_2 \quad (14)$$

Statement (14) is true because of $s_1 \cong s_2 \implies \text{den}(s_1) =_I \text{den}(s_2) \implies s_1 =_I s_2$. Thus for anonymous signs equality and equinymy coincide. It is of interest to consider interpretants in which equinymy and synonymy coincide. That means that synonyms have equal instead of just synonymous denotations. This leads to the next definition:

A set I of compatible interpretants is called mergeable iff for all signs in I

$$s_1 \equiv_I s_2 \implies s_1 \cong_I s_2 \quad (15)$$

which means that all of its synonyms are equinymy. If an interpretant is not mergeable with itself it can usually be split into several different interpretants which are mergeable with themselves. In mergeable interpretants, it follows that

$$s_1 =_I s_2 \iff s_1 \dot{=} s_2 \implies s_1 \cong_I s_2 \iff s_1 \equiv_I s_2 \quad (16)$$

$$s_1 =_I s_2 \iff \text{rmn}(s_1) = \text{rmn}(s_2) \quad (17)$$

because $\text{rmn}(s_1) = \text{rmn}(s_2) \implies \text{dmn}(s_1) \equiv_I \text{dmn}(s_2) \implies \text{dmn}(s_1) \dot{=} \text{dmn}(s_2)$.

From (14) and (16) it follows that for anonymous signs in mergeable interpretants, the four equivalences, synonymy, polysemy, equality and equinymy are all the same. For anonymous signs in a mergeable interpretant, the representamen rules alone determine synonymy. Because representamens are mathematical entities, it follows that anonymous signs in mergeable interpretants behave like mathematical entities.

6 Conceptual Structures

While semiotic structures, such as synonymy, model the decisions a user makes with respect to a formal language, mathematical entities can be used to compute the consequences of such decisions. It is argued in this paper that the mathematical entities involved in signs (especially formal concepts and contexts) can be modelled as conceptual structures using formal concept analysis. Concept lattices can be used to show the consequences of the semiotic design decisions. Users can browse through concept lattices using a variety of existing software tools to explore the signs.

This insight is not new. In fact there are several papers, (for example, Snelting (1996)) which demonstrate the usefulness of formal concept analysis in software engineering. Our semiotic conceptual framework adds a layer of explanation to these studies by detailing how semiotic and conceptual aspects both contribute to formal languages. A semiotic perspective also adds modes of communication or “speech acts” to the conceptual framework. Examples are “assertion”, “query” and “question”. But these are not further discussed in this paper.

Formal concept analysis, description logics, Sowa’s (1984) conceptual graphs, Barwise & Seligman’s (1997) classifications and object-oriented formalisms each provide a slightly different definition of concepts. But they all model denotations as binary relations of types and instances/values. Thus denotations are signs of the form $[\text{typ}(s) : \text{ins}(s)]$ where $\text{typ}(s)$ is a sign called type and $\text{ins}(s)$ is a sign called instance (or value). A sign s with $\text{den}(s) \cong_I [\text{typ}(s) : \text{ins}(s)]$ is written as $s[\text{typ}(s) : \text{ins}(s)]$. A formal concept is an anonymous sign $c =_I (\{e_1, e_2, \dots\}; \{i_1, i_2, \dots\})$ where e_1, e_2, \dots are mathematical entities that form the extension and i_1, i_2, \dots are mathematical entities that form the intension of the formal sign.

For a fixed interpretant, denotations are mapped onto formal concepts as follows: the intension is the set of types that are inherited by the sign via a type hierarchy and the extension is the set of instances that share exactly those types. As a restriction it should be required that each sign is synonymous to at most one formal concept within the given interpretant. Within the framework of formal concepts, equality of denotations can be mathematically evaluated because formal concepts are anonymous signs. Formal concepts as defined in this paper form concept lattices as defined in formal concept analysis.

7 Contexts and Meta-constructs

Several formalisms for contexts have been suggested by AI researchers (eg. McCarthy (1993)) but in general they are not integrated into reasoning applications as frequently and not as well understood as representamens and formal concepts. Figure 1 indicates that contexts should play an important role in the formalisation of signs besides representamens and formal concepts. We argue in this paper, that contexts are not as difficult to deal with as AI research suggests if they are modelled as formal concepts as well.

If contexts are modelled as formal concepts, relationships between contexts, such as containment, temporal and spatial adjacency or overlap can be modelled as conceptual relations. Contexts as formal concepts are denotations of other signs with respect to other interpretants. Peirce stresses the existence of infinite chains of interpretants (interpretants of interpretants of interpretants ...). But as formal concepts, contexts are not modelled as contexts of contexts of contexts. All contexts can be modelled as formal concepts with respect to one special meta-context of contexts because containment chains are just conceptual relations, not meta-relations.

An advantage of this approach is that apart from one meta-language which describes the semiotic conceptual framework, no other meta-languages are required. All other seemingly “meta”-languages are modelled via conceptual containment relations between their corresponding contexts. For example, all facts, rules and constructors of a programming language can be described in a single context. A program of that language is executed in a different context. Both contexts are related via a containment relation with respect to the meta-context of contexts. But the context of a programming language is not a meta-context of a program.

8 Examples

The condition of mergeability of interpretants states that synonymous signs must have equal denotations. With respect to programming languages this means that as soon as a variable changes its value, a new interpretant must be formed. It may be possible to bundle sequential changes of values. For example, if all values in an array are updated sequentially, it may be sufficient to assume one interpretant before the changes and one after the changes instead of forming a separate interpretant after each change. Some variables may also be ignored, such as counters in for-statements. This corresponds to the distinction between persistent and transient objects in object-oriented modelling. Transient variables do not initiate new interpretants.

```

counter = 1
print "game starts"
while counter <= 5:
    number = input("please guess the number")
    if number == 5:
        print "good guess"
        break
    else:
        print "try again"
    counter = counter + 1
else:
    print "game over"

```

Fig. 2. A piece of Python code

The significance of the following examples is not that this is just another application of formal concept analysis but instead that this kind of analysis is suggested by the semiotic conceptual framework. The theory about mergeability of interpretants suggests that a number of different interpretants are invoked by any computer program depending on when certain variables change their values. It just happens that formal concept analysis can be used to analyse this data. We believe that careful consideration of the predictions made by the semiotic conceptual framework can potentially provide interesting insights. But we have not yet explored this further.

The example in figure 2 shows a piece of Python code and a corresponding concept lattice in figure 3. The contexts (or formalisable parts of interpretants) are initiated by the changes of the variables “counter” and “number”. Each context produces a different behaviour, i.e., a different print statement by the program. The lattice in figure 3 is modelled as follows: the objects are the observable behaviours of the program (i.e., the print statements). The attributes are the states of the variables which are relevant for the contexts. The formal concepts are contexts of the program. If the counter is smaller than 5 and the user guesses the number 5, the program prints “good guess”. If the number is not 5 but the counter is smaller than 5, the program prints “try again” and “please guess”, except in the first instance (counter = 1) when it prints “please guess” after having printed “game starts”. If the counter is larger than 5 the game prints “game over”.

In contrast to flowcharts, the lattice representation does not represent the sequence of the statements. Wolff & Yameogo’s (2003) temporal concept analysis could be applied

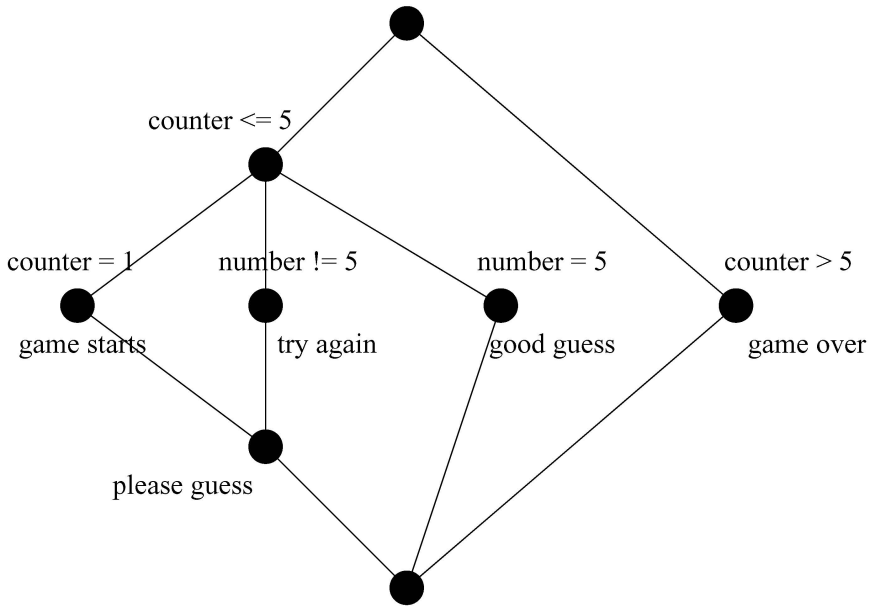


Fig. 3. A lattice of contexts for the code in figure 2

to the lattice to insert the temporal sequence. The lattice shows the relationships among the contexts. For example, it shows that the start-context ($\text{counter} = 1$) and the contexts in which the wrong number was guessed share behaviour. This is not necessarily clearly expressed in the code itself. In fact our experience with teaching scripting languages to non-programmers has shown that students often have a problem comprehending where the 'print "please guess"' statement needs to be placed within the while loop so that it pertains to both the first iteration and to some of the later iterations. In the lattice this relationship is shown more clearly.

It should be noted that we have not yet tested whether students can read the lattices. But we are not suggesting that lattices must be used directly as a software engineering tool. The information contained in the lattice could be displayed in a different format, which would still need to be determined. We have also not yet determined in how far such lattices can be automatically generated from code. We intend to investigate this further in the near future.

The second example, which was taken from Ballentyne (1992) demonstrates the equivalence of Warnier diagrams (Orr, 1977) and concept lattices. The original data is shown in the upper left table in figure 4 and to be read as follows: action 1 has four crosses which correspond to the conditions $A, \neg B, \neg C$ or $A, \neg B, C$ or $A, B, \neg C$ or A, B, C . This is equivalent to condition A implying action 1. Therefore in the formal context on the left there is a cross for A and 1. Action 2 is conditioned by $\neg A$ and B which is visible both in the left table and in the formal context. After the whole context has been constructed in that manner, a second condition is to be considered which is that A and $\neg A$ and so on must exclude each other. This is true for A and B but $\neg C$ does not have

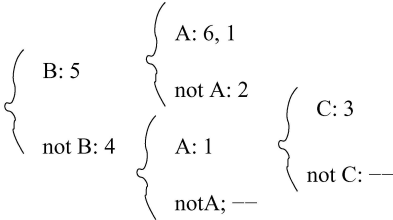
Condition/action list

Conditions A B C	Actions					
	1	2	3	4	5	6
O O O				x		
O O 1				x		
O 1 O		x			x	
O 1 1		x			x	
1 O O	x			x		
1 O 1	x		x	x		
1 1 O	x				x	x
1 1 1	x				x	x

Corresponding formal context

	1	2	3	4	5	6	t
A	x		x			x	x
B		x			x	x	
C			x				
not A		x					
not B			x	x			x
not C							

Warnier diagram



Concept lattice

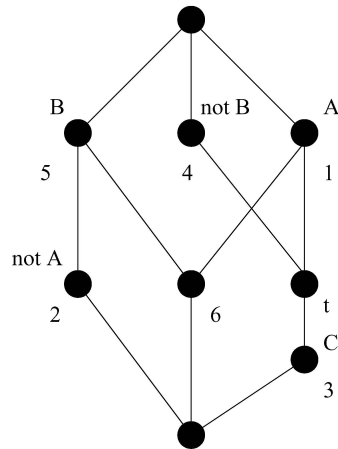


Fig. 4. Warnier diagrams and lattices

any attributes and must be excluded from the lattice. A third condition is that any meet irreducible concepts in the lattice must not be labelled by an attribute because otherwise that attribute would be implied by other attributes without serving as a condition itself. For this reason, the temporary attribute t is inserted. The resulting lattice can be read in the same manner as the one in figure 3. The Warnier diagram corresponds to a set of paths from the top to the bottom of the lattice which cover all concepts. Obviously, there can be different Warnier diagrams corresponding to the same lattice.

9 Conclusion

A semiotic conceptual framework for formal languages combines conceptual reasoning and inference structures with semiotic modes, such as assertion, question and query. By considering the denotations of formal signs as formal concepts, structure is imposed.

Because denotations are both signs and can be mapped to formal concepts which are mathematical entities, denotations serve as boundary objects (Star, 1989) between the mathematical world and the pragmatic world of signs. The role of contexts is often neglected. This is understandable in mathematical applications because mathematical entities exist in more global contexts. But in other formal languages, which employ richer signs, contexts are frequently changing and cannot be ignored. If contexts are modelled as formal concepts, it is not necessary to invent any new structures but instead the mechanisms of formal concept analysis can also be applied to contexts. Contexts provide a means for managing signs and sign relations. Programming languages and databases already fulfill these functions, but so far not much theory has been developed which explains the theoretical foundations of context management. A semiotic conceptual framework can provide such a theory.

References

1. Ballentyne, George (1992). Class notes. Unpublished manuscript.
2. Barwise, Jon; Seligman, Jerry (1997). *Information Flow*. The Logic of Distributed Systems. Cambridge University Press.
3. Ganter, B.; Wille, R. (1999). *Formal Concept Analysis*. Mathematical Foundations. Berlin-Heidelberg-New York: Springer, Berlin-Heidelberg.
4. McCarthy, John (1993). *Notes on Formalizing Context*. Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France. p. 555-560.
5. Orr, K. T. (1977). *Structured Systems Development*. Yourdon Press, New York.
6. Peirce, Charles (1897). *A Fragment*. CP 2.228. In: Collected papers. Hartshorne & Weiss. (Eds. Vol 1-6); Burks (Ed. Vol 7-8), Cambridge, Harvard University Press, 1958-1966.
7. Prechelt, Lutz (2000). *An empirical comparison of C, C++, Java, Perl, Python, Rexx, and TCL*. IEEE Computer, 33(10), p. 23-29.
8. Snelting, Gregor (1995). *Reengineering of Configurations Based on Mathematical Concept Analysis*. ACM Transactions on Software Engineering and Methodology 5, 2, p. 99-110.
9. Sowa, J. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA.
10. Star, Susan Leigh (1989). *The structure of ill-structured solutions: Boundary objects and heterogeneous problem-solving*. In: Gasser & Huhns (Eds). Distributed Artificial Intelligence, Vol 2, Pirman, p. 37-54.
11. Wolff, Karl Erich; Yameogo, Wendsomde (2003). *Time Dimension, Objects, and Life Tracks. A Conceptual Analysis*. In: de Moor; Lex; Ganter (Eds.). Conceptual Structures for Knowledge Creation and Communication. Lecture notes in Ai 2746. Springer Verlag.

A Mathematical Model for TOSCANA-Systems: Conceptual Data Systems

Joachim Hereth Correia and Tim B. Kaiser

Darmstadt University of Technology
Schlossgartenstr. 7, 64289 Darmstadt, Germany
{hereth,tkaiser}@mathematik.tu-darmstadt.de

Abstract. The development of the theory of Formal Concept Analysis has been accompanied from its beginning by applications of the theory to real-world problems. Those applications gave rise to the implementation of the software TOSCANA and the creation of TOSCANA-systems. In this paper, we provide a mathematical model for these systems. This model – called *Conceptual Data System* – enables us to describe TOSCANA-systems and to discuss possible extensions in mathematical terminology.

1 Introduction

Since its beginning the development of the mathematical theory of Formal Concept Analysis has been accompanied not only by theoretical considerations but also by practical applications of the theory to real-world problems. During the last decade, one particular group of projects contributed greatly to the deployment of Formal Concept Analysis. TOSCANA-systems – based on the software TOSCANA – showed how the theory can be used for instance for analyzing and restructuring data or to retrieve documents from a given database. The main tasks of the software itself is to visualize diagrams, to calculate the realized extents of the displayed conceptual scales and to allow the user to select and to construct the view on the data he is interested in. While TOSCANA is for the most part an implementation of the theoretical concepts of conceptual scaling (cf. [GW89]), TOSCANA-systems themselves have developed a rich structure which makes them flexible and adaptable to many different situations. While parts of this structure have been described already very early in [VWW91, SSVWW93], a complete mathematical description of the system and its interface is not available. Our approach tries to combine the previous work and extend it using some ideas from one of the authors diploma thesis [Ka02] to provide a formal basis for discussion about the structure, development, and extension of TOSCANA-systems. We consider a mathematical model for those systems to be helpful to transport mathematical development to the applied side, and vice-versa to translate problems arising in real-world projects back to the theoretic level.

2 Many-Valued Context and Conceptual Schema

A TOSCANA-system implements the idea of conceptual scaling [GW89] where a data table is modeled as a *many-valued context*:

Definition 1 (many-valued context). A many-valued context is a structure $\mathbb{K} := (G, M, \bigcup_{m \in M} W_m, I)$, where G is a set of objects, M is a set of attributes, $W := \bigcup_{m \in M} W_m$ is a set of values and $I \subseteq G \times M \times W$ is a ternary relation, where $(g, m, w_1), (g, m, w_2) \in I \Rightarrow w_1 = w_2 \in W_m$. Every $m \in M$ can be understood as a (partial) function from G to W with $m(g) := w$ if $(g, m, w) \in I$. By W_m we denote the set of potential values for an attribute m , while $m(G)$ is the set of actual values occurring in the context.

A *conceptual scale* for a set of attributes of a many-valued context is defined as follows:

Definition 2 (conceptual scale). Let $\mathbb{K} := (G, M, W, I)$ be a many-valued context and $N \subseteq M$. Then we call a formal context $\mathbb{S}_N := (G_N, M_N, I_N)$ conceptual scale if $\{(m(g))_{m \in N} \mid g \in G\} \subseteq G_N \subseteq \times_{m \in N} W_m$. We say \mathbb{S}_N scales the attribute set N . A family of conceptual scales $(\mathbb{S}_{N_j})_{j \in J}$ scales \mathbb{K} if every conceptual scale \mathbb{S}_{N_j} scales the attribute set $N_j \subseteq M$.

Central to all TOSCANA-systems are the diagrams that are used as interface to the data. To the user, the conceptual scale and the corresponding diagram appear as one entity. Mathematically however, we differentiate between the already introduced conceptual scale and its geometrical representation.

Definition 3 (diagram map). If (P, \leq) is an ordered set, P is finite, and \prec denotes the lower neighbour relation for \leq , we call an injective mapping

$$\lambda : P \cup \prec \rightarrow \mathbb{R}^2 \cup \mathfrak{P}(\mathbb{R}^2)$$

diagram map if

- $p \in P \Rightarrow \lambda(p) \in \mathbb{R}^2$,
- $p_1 < p_2 \Rightarrow \lambda(p_1)|_2 < \lambda(p_2)|_2$, and
- $(p_1, p_2) \in \prec \Rightarrow \lambda(p_1, p_2) := \{r\lambda(p_1) + (1-r)\lambda(p_2) \mid r \in \mathbb{R} \text{ and } 0 \leq r \leq 1\}$.

The image $\lambda(P \cup \prec)$ of a diagram map represents a line diagram of the ordered set (P, \leq) . The image $\lambda(P)$ is the set of all *points* and $\lambda(\prec)$ is the set of all *line segments* of the line diagram. If $P = \mathfrak{B}(\mathbb{K})$, we can assign a labeling $(G_c, M_c)_{c \in \mathfrak{B}(\mathbb{K})}$ where $(G_c, M_c) \in \mathfrak{P}(G_{\mathbb{K}}) \times \mathfrak{P}(M_{\mathbb{K}})$ to a diagram map λ . The labels can be attached to the corresponding points $v \in \lambda(\mathfrak{B}(\mathbb{K}))$ using λ . The label (G_c, M_c) is attached to the point $\lambda(c)$. A simple way to assign a labeling to λ is $(c)_{c \in \mathfrak{B}(\mathbb{K})}$. In TOSCANA-systems it is common to label attributes reduced, i. e. to list only contingents. With γ we refer to the object concept mapping from the Basic Theorem on Formal Concept Analysis and with μ to the attribute concept mapping. Then we define

$$(\text{Ext}(c), \mu^{-1}(c))_{c \in \mathfrak{B}(\mathbb{K})}$$

as *complete labeling*. The *reduced labeling* is defined as

$$(\gamma^{-1}(\mathbf{c}), \mu^{-1}(\mathbf{c}))_{\mathbf{c} \in \mathfrak{B}(\mathbb{K})}.$$

The process of developing a TOSCANA-system is an iterative interdisciplinary task where a discussion between domain experts and Formal Concept Analysis experts yield conceptual scales for a database of interest. The result of this process is a *conceptual schema* which will be defined mathematically in the following.

Definition 4 (conceptual schema). *Let $(\mathbb{S}_{N_j})_{j \in J}$ be a family of conceptual scales and let $(\lambda_j)_{j \in J}$ be a family of diagram maps where $\text{dom}(\lambda_j) = \mathfrak{B}(\mathbb{S}_{N_j}) \cup \prec_j$. Then we call the vector $\mathcal{S} := (\mathbb{S}_{N_j}, \lambda_j)_{j \in J}$ conceptual schema. We say that a conceptual schema \mathcal{S} and a many-valued context \mathbb{K} are consistent if $(\mathbb{S}_{N_j})_{j \in J}$ scales \mathbb{K} .*

Our formalization of the conceptual schema adapts the conceptual file model from [VWW91] (the term was changed to *conceptual schema* since the information is not necessarily stored in a single file). The information necessary to connect the database is very subtle modelled by the sets $(N_j)_{j \in J}$. This reflects the fact that in TOSCANA-systems one cannot change the column names in a data table without adapting the queries for the diagrams. Note that the object set of a conceptual scale is contained in $\bigtimes_{m \in N_j} W_m$ according to Definition 2. Therefore the labeling for a diagram map λ_j contains in the object part tuples of potential attribute values.

With the preceding definitions we have formalized the basic ingredients of a conceptual data system. Next, we will discuss a description of the interface.

3 Conceptual Interface and Conceptual Data System

With the notion *conceptual interface* we will construct a mathematization of the interface of the system, which is usually provided by the TOSCANA-software. The TOSCANA-software uses the information from the real-world counterpart of a conceptual schema to present the data to the user, and it allows the user to interact with the system. A conceptual interface, a conceptual schema, and a many-valued context will form the components of a *conceptual data system*. The following definitions will be put together in the end. We start describing the connection between many-valued context and conceptual scales.

Definition 5 (realized scale). *Let $\mathcal{S} := (\mathbb{S}_{N_j}, \lambda_j)_{j \in J}$ be a conceptual schema consistent with $\mathbb{K} := (G, M, W, I)$ and let $j \in J$. For $\mathbb{S}_{N_j} := (G_{N_j}, M_{N_j}, I_{N_j})$ the realized scale is defined as $\mathbb{S}_{N_j}^r := (G, M_{N_j}, I_{N_j}^r)$ with $(g, m) \in I_{N_j}^r : \iff ((n(g))_{n \in N_j}, m) \in I_{N_j}$.*

It is important to note that an object $h \in G_{N_j}$ can be *non-realized* if there is no $g \in G$ meeting the attribute value combination prescribed by h . The object clarification of a realized scale $\mathbb{S}_{N_j}^r$ is isomorphic to the subcontext $\hat{\mathbb{S}}_{N_j} := (\hat{G}_{N_j}, M_{N_j}, I_{N_j} \cap (\hat{G}_{N_j} \times M_{N_j}))$ of \mathbb{S}_{N_j} with

$$\hat{G}_{N_j} := \{h \in G_{N_j} \mid \exists g \in G : (m(g))_{m \in N_j} = h\}.$$

Therefore, we can embed the lattice $\underline{\mathfrak{B}}(\mathbb{S}^r)$ in $\underline{\mathfrak{B}}(\mathbb{S})$, identifying concepts by their intents, using the following theorem from [GW99, p. 98]:

Theorem 6. *For $H \subseteq G$, the map*

$$\begin{aligned} \beta : \underline{\mathfrak{B}}(H, M, I \cap H \times M) &\longrightarrow \underline{\mathfrak{B}}(G, M, I) \\ (A, B) &\longmapsto (B', B) \end{aligned}$$

is a \vee -preserving order-embedding.

If this embedding β is not surjective, *non-realized concepts* will occur. These are concepts of \mathbb{S} which have no preimage under the natural embedding β . Let $\mathfrak{c} := (A, B)$ be in $\underline{\mathfrak{B}}(\mathbb{S})$. One can check whether \mathfrak{c} is non-realized by deriving B^{I_r} . If (B^{I_r}, B) is not an element of $\underline{\mathfrak{B}}(\mathbb{S}^r)$ the concept \mathfrak{c} is non-realized.

Definition 7 (non-realized concept). *We call a concept $\mathfrak{c} \in \underline{\mathfrak{B}}(\mathbb{S})$ non-realized in $\underline{\mathfrak{B}}(\mathbb{S}^r)$ if $\mathfrak{c} \notin \beta(\underline{\mathfrak{B}}(\mathbb{S}^r))$.*

If a conceptual scale \mathbb{S}_N scales an attribute set N of a many-valued context \mathbb{K} and for every $h \in G_N$ there exists an $g \in G$ with $(m(g))_{m \in N} = h$, we call \mathbb{S}^r *completely realized*. Discussions of non-realized concepts and their relevance for *local scaling* can be found in [St96, Sc98]. If a conceptual schema \mathcal{S} and a many-valued context \mathbb{K} are consistent we can define additional notions of labelings. Let $\mathbb{S}_{N_j} := (G_{N_j}, M_{N_j}, I_{N_j})$ be a scale of a conceptual schema \mathcal{S} . Realized labelings for the corresponding diagram map $\lambda_j : \underline{\mathfrak{B}}(\mathbb{S}_{N_j}) \cup \prec \rightarrow \mathbb{R}^2 \cup \mathfrak{P}(\mathbb{R}^2)$ are subsets of $\mathfrak{P}(G) \times \mathfrak{P}(M_{N_j})$. The *complete realized labeling* is defined as

$$((\text{Int}(\mathfrak{c}))^{I_{N_j}^r}, \mu^{-1}(\mathfrak{c}))_{\mathfrak{c} \in \underline{\mathfrak{B}}(\mathbb{S}_{N_j})}$$

and the *reduced realized labeling* is defined as

$$((\text{Int}(\mathfrak{c}))^{I_{N_j}^r} \setminus \bigcup_{\mathfrak{c}' < \mathfrak{c}} (\text{Int}(\mathfrak{c}'))^{I_{N_j}^r}, \mu^{-1}(\mathfrak{c}))_{\mathfrak{c} \in \underline{\mathfrak{B}}(\mathbb{S}_{N_j})}.$$

3.1 Zooming

In a TOSCANA-system, the user can select a subset of the available objects for further analysis by double-clicking on a given concept. This is called *zooming*. More precisely, the user changes the state of the system into another state, enabling the system to produce a new diagram. The *state* of a TOSCANA-system is given by the selected scales and the chosen object sets for filtering. We formally define:

Definition 8 (state, initial state). *Let $\mathcal{S} := (\mathbb{S}_{N_j}, \lambda_j)_{j \in J}$ be a conceptual schema consistent with a many-valued context $\mathbb{K} := (G, M, W, I)$. Then a state is a triple $\mathbf{s} := (\sigma, F_1, F_2)$, where $F_1 \subseteq F_2 \subseteq G$ and $\sigma := (j_i)_{i=1}^n$ with $j_i \in J$ for $i \in \{1, \dots, n\}$ and $i \neq k \Rightarrow j_i \neq j_k$. F_1 is called *exact zooming filter* and F_2 is called *full zooming filter*. An *initial state* is a state where $F_1 = F_2 = G$.*

The scales identified by σ are those that are displayed to the user. The number of these *active scales* coincides with the depth of nesting n . To allow the user to switch between exact and full zooming filter, we maintain both sets, F_1 and F_2 .

Formally, *zooming* maps states to states, depending on the user input which consists of a concept and a next diagram to zoom into. In a real application this can be a mouse click on a node of the line diagram displayed and a choice of a certain number of diagrams of interest, for instance the next diagram to zoom into.

We need to recall a definition from [GW99]:

Definition 9. We call $\mathbb{K}_1 | \mathbb{K}_2 := (G_1, M_1 \dot{\cup} M_2, I_1 \dot{\cup} I_2)$ the *apposition* of \mathbb{K}_1 and \mathbb{K}_2 .

Definition 10. We define *zooming* as the mapping

$$\zeta : (\mathbf{s}_1, \mathbf{c}, j) \mapsto \mathbf{s}_2$$

where $\mathbf{s}_1 := (\sigma, F_1, F_2)$ and $\mathbf{s}_2 := (\sigma', F'_1, F'_2)$ are states, $j \in J$, and $\mathbf{c} \in \mathfrak{B}(|_{i \in \{1, \dots, n\}} \mathbb{S}_{j_i}^r)$. Then

$$F'_1 := F_1 \cap \gamma^{-1}(\mathbf{c}),$$

and

$$F'_2 := F_2 \cap \text{Ext}(\mathbf{c}).$$

For $\sigma := (j_i)_{i=1}^n$, the new set σ' of active scales is $\sigma' := (j_i)_{i=2}^{n+1}$ where $j_{n+1} = j$.

We call a state *valid* if it is an initial state or the result of the zooming operation with a valid state as input. After having formalized zooming, we turn to the more sophisticated (nested) diagram display.

3.2 Diagram Display

An important aspect for TOSCANA-systems is the feature to combine several predefined diagrams into one more complex view onto the data by nested line diagrams. To describe this mathematically, we introduce an operation between diagram maps:

Definition 11 (\odot). Let (P_1, \leq_1) and (P_2, \leq_2) be finite ordered sets with lower neighbour relations \prec_1 and \prec_2 ; furthermore let \prec_{12} be the lower neighbour relation of the direct product $(P_1, \leq_1) \times (P_2, \leq_2)$. For a diagram map λ_1 of (P_1, \leq_1) and a diagram map λ_2 of (P_2, \leq_2) , positive reals s, r_1 , and r_2 can be chosen such that $r_1 \min\{\|v - w\| \mid v, w \in \lambda_1(P_1)\} > s$ and $r_2 \max\{\|v\| \mid v \in \lambda_2(P_2)\} \leq s$. For such real numbers, a diagram map $\lambda := \lambda_1 \odot \lambda_2 : P_1 \times P_2 \cup \prec_{12} \rightarrow \mathbb{R}^2 \cup \mathfrak{P}(\mathbb{R}^2)$ exists with

$$\lambda(p_1, p_2) := r_1 \lambda_1(p_1) + r_2 \lambda_2(p_2) \text{ and}$$

$$\lambda((p_1, p_2), (q_1, q_2)) := \{r(p_1, p_2) + (1 - r)(q_1, q_2) \mid r \in \mathbb{R} \text{ with } 0 \leq r \leq 1\}.$$

Proposition 12. The class of diagram maps is closed under the operations \odot defined in Definition 11.

Proof. Let $\lambda := \lambda_1 \odot \lambda_2$. We have to show

$$(p_1, p_2) < (q_1, q_2) \implies \lambda(p_1, p_2)|_2 < \lambda(q_1, q_2)|_2.$$

Assume $(p_1, p_2) < (q_1, q_2)$. Then

$$\begin{aligned} \lambda(p_1, p_2)|_2 &= (r_1 \lambda_1(p_1) + r_2 \lambda_2(p_2))|_2 = r_1 \lambda_1(p_1)|_2 + r_2 \lambda_2(p_2)|_2 \\ &< r_1 \lambda_1(q_1)|_2 + r_2 \lambda_2(q_2)|_2 = (r_1 \lambda_1(q_1) + r_2 \lambda_2(q_2))|_2 = \lambda(q_1, q_2)|_2. \end{aligned}$$

It remains to show that λ is injective. If $\lambda(p_1, p_2) = \lambda(q_1, q_2)$, then

$$r_1 \lambda_1(p_1) + r_2 \lambda_2(p_2) = r_1 \lambda_1(q_1) + r_2 \lambda_2(q_2)$$

which can be transformed to

$$r_1(\lambda_1(p_1) - \lambda_1(q_1)) = r_2(\lambda_2(p_2) - \lambda_2(q_2)).$$

This implies

$$r_1 \|\lambda_1(p_1) - \lambda_1(q_1)\| = r_2 \|\lambda_2(p_2) - \lambda_2(q_2)\|.$$

If $p_1 \neq q_1$, the injectivity of λ_1 and the choice of r_1 force the left hand side of the above equation to be greater than s . Therefore $p_2 \neq q_2$, because otherwise the right hand side would be 0. The choice of r_2 implies that the right hand side of the above equation is less or equal than s . It follows that $p_1 = q_1$ which implies $p_2 = q_2$, because λ_2 is injective. \square

From the viewpoint of Formal Concept Analysis a TOSCANA-system visualizes the concept lattice of the *apportion* of the participating realized scales embedded into the direct product of the concept lattices of the corresponding conceptual scales. Because the object set of different realized scales of the same conceptual schema is equal, we maintain the object set and form the disjoint union of attributes and incidence relations for combining two scales. Proposition 31 and Theorem 7 from [GW99, p. 98, p. 77], guarantee that an order embedding of $\mathfrak{B}(\mathbb{S}_1^r | \mathbb{S}_2^r)$ into $\mathfrak{B}(\mathbb{S}_1) \times \mathfrak{B}(\mathbb{S}_2)$ is always possible. After summarizing the theoretical background, we give the definition for a diagram display.

Definition 13 (diagram display). Let \mathcal{S} be a conceptual schema consistent with a many-valued context \mathbb{K} and \mathbf{s} be a valid state of the former. A diagram display is a mapping

$$\delta : (\mathbb{K}, \mathcal{S}, \mathbf{s}) \longmapsto \lambda$$

with $\lambda := \odot_{i=1}^n \lambda_{j_i}$.

We have to explain how a diagram map resulting of the operation \odot can be labeled. Let λ_1 and λ_2 be diagram maps with $P_1 := \mathfrak{B}(\mathbb{S}_1)$ and $P_2 := \mathfrak{B}(\mathbb{S}_2)$. For $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathfrak{B}(\mathbb{S}_1) \times \mathfrak{B}(\mathbb{S}_2)$ we abbreviate $\text{Int}(\mathbf{c}_1) \cup \text{Int}(\mathbf{c}_2)$ by writing $\text{Int}(\mathbf{c})$ and $\mu^{-1}(\mathbf{c}_1) \cup \mu^{-1}(\mathbf{c}_2)$ by writing $\mu^{-1}(\mathbf{c})$. Let $\mathbf{s} := (\sigma, F_1, F_2)$ be a valid state of the system. Then we can distinguish four different labelings. In the following

$I_{1,2}^r$ denotes the incidence relation of the formal context $\mathbb{S}_1^r | \mathbb{S}_2^r$. The *complete s-realized labeling* is defined as

$$((F_2 \cap \text{Int}(\mathbf{c})^{I_{1,2}^r}, \mu^{-1}(\mathbf{c}))_{\mathbf{c} \in \mathfrak{B}(\mathbb{S}_1) \times \mathfrak{B}(\mathbb{S}_2)}.$$

This labeling corresponds to the setting *all documents* in TOSCANA 2 and TOSCANA 3 and to the *Show all matches - Filter: use all matches* setting in TOSCANA.J. If we use a reduced object labeling, we get the *special s-realized labeling*

$$((F_2 \cap (\text{Int}(\mathbf{c})^{I_{1,2}^r} \setminus \bigcup_{\mathbf{c}' < \mathbf{c}} (\text{Int}(\mathbf{c}')^{I_{1,2}^r}), \mu^{-1}(\mathbf{c}))_{\mathbf{c} \in \mathfrak{B}(\mathbb{S}_1) \times \mathfrak{B}(\mathbb{S}_2)}.$$

This labeling corresponds to the setting *special documents* in TOSCANA 3 and to *Show only exact matches - Filter: use all matches* in TOSCANA.J. If we now apply the smaller filter F_1 we get a *exact s-realized labeling*, which corresponds to the setting *exact documents* in TOSCANA 3 and to *Show only exact matches - Filter: use only exact matches* in TOSCANA.J:

$$((F_1 \cap (\text{Int}(\mathbf{c})^{I_{1,2}^r} \setminus \bigcup_{\mathbf{c}' < \mathbf{c}} (\text{Int}(\mathbf{c}')^{I_{1,2}^r}), \mu^{-1}(\mathbf{c}))_{\mathbf{c} \in \mathfrak{B}(\mathbb{S}_1) \times \mathfrak{B}(\mathbb{S}_2)}.$$

The fourth combinatorial possibility is seldom used in TOSCANA systems, but we mention it here for the sake of completeness:

$$((F_1 \cap \text{Int}(\mathbf{c})^{I_{1,2}^r}, \mu^{-1}(\mathbf{c}))_{\mathbf{c} \in \mathfrak{B}(\mathbb{S}_1) \times \mathfrak{B}(\mathbb{S}_2)}.$$

Constructing the object parts of the labeling by deriving the concept intents via the relation $I_{1,2}^r$, we obtain an order embedding of the concept lattice of the apposition of the realized scales into the direct product of the concept lattices of the conceptual scales. The derivation operator $J_{1,2}^r : \mathfrak{P}(M_1 \cup M_2) \longrightarrow \mathfrak{P}(G)$ models the database queries.

3.3 Final Formalizations

Now we can give a compact definition of what can be understood as a conceptual interface:

Definition 14 (conceptual interface). Let \mathbb{K} be a many-valued context consistent with the conceptual schema \mathcal{S} . A conceptual interface is a pair $\mathcal{I} := (J^r, \Phi)$ such that for every subset L of the index set K of the conceptual schema \mathcal{S} , the operator J_L^r is the derivation operator for the formal context $|_{i \in L} \mathbb{S}_i^r$ and the set Φ contains at least the essential mappings δ for diagram display and ζ for the zooming operation.

The next definition puts all the parts together.

Definition 15 (conceptual data system). Let \mathbb{K} be a many-valued context consistent with the conceptual schema \mathcal{S} and \mathcal{I} a conceptual interface. Then $\mathcal{CDS} := (\mathbb{K}, \mathcal{S}, \mathcal{I})$ is called a conceptual data system.

4 Conclusion

We have formalized the core part of all existing TOSCANA-systems by the notion of a conceptual data system. With this formalization we provide means for describing the interface of a TOSCANA-system formally as well as its interaction with the conceptual schema and the database. We give mathematical descriptions for the different labeling methods and summarize the mathematical theorems necessary to illuminate the conceptual meaning of the diagrams. This framework may help in investigating and discussing different extensions for TOSCANA-systems mathematically, for instance in course of the TOSCANAJ project (cf. [BH03]).

References

- [BH03] P. Becker, J. Hereth Correia: The Toscanaj Suite for Implementing Conceptual Information Systems. Preprint 2003.
- [GW89] B. Ganter, R. Wille: Conceptual Scaling. In: F. Roberts (ed.): Applications of combinatorics and graph theory to the biological and social sciences. Springer, Berlin – Heidelberg – New York 1989, 139–167.
- [GW99] B. Ganter, R. Wille: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin – Heidelberg – New York 1999.
- [Ka02] T. B. Kaiser: Conceptual Data Systems – Providing a Mathematical Basis for TOSCANA-systems. Diplomarbeit, TU Darmstadt 2002.
- [Sc98] A. Schatz: Lokales Skalieren, Mathematische Grundlagen und objektorientiertes Design. Diplomarbeit, TH Darmstadt 1998.
- [SSVWW93] P. Scheich, M. Skorsky, F. Vogt, C. Wachter, R. Wille: Conceptual Data Systems. In: O. Opitz, B. Lausen, R. Klar (eds.): Information and classification. Springer, Berlin – Heidelberg 1993, 72–84.
- [St96] G. Stumme: Local scaling in conceptual data systems. In: P.W. Eklund, G. Ellis, G. Mann (eds.): Conceptual Structures: Knowledge representation as interlingua. LNAI 1115. Springer, Berlin – Heidelberg 1996, 308–320.
- [VWW91] F. Vogt, C. Wachter, R. Wille: Data analysis based on a conceptual file. In: H.H. Bock and P. Ihm (eds.): Classification, data analysis and knowledge organization. Springer, Berlin – Heidelberg 1991, 131–142.

BLID: An Application of Logical Information Systems to Bioinformatics*

Sébastien Ferré and Ross D. King

Department of Computer Science, University of Wales, Aberystwyth
Penglais, Aberystwyth SY23 3DB, UK
{sbf,rdk}@aber.ac.uk

Tel: +44 1970 621922, Fax: +44 1970 622455

Abstract. BLID (Bio-Logical Intelligent Database) is a bioinformatic system designed to help biologists extract new knowledge from raw genome data by providing high-level facilities for both data browsing and analysis. We describe BLID's novel data browsing system which is based on the idea of Logical Information Systems. This enables combined querying and navigation of data in BLID (extracted from public bioinformatic repositories). The browsing language is a logic especially designed for bioinformatics. It currently includes sequence motifs, taxonomies, and macromolecule structures, and it is designed to be easily extensible, as it is composed of reusable components. Navigation is tightly combined with this logic, and assists users in browsing a genome through a form of human-computer dialog.

1 Motivation

Over the last decade many organisms have had their genomes fully sequenced. For example, the 17 chromosomes of the Baker's Yeast (*Saccharomyces Cerevisiae*) have been sequenced, and they code for about 6000 proteins [Gof97]. Yeast is one of the best studied of all organisms, yet about 30% of all its proteins still have not yet any known function. For other organism the percentage is higher. Therefore, one of most important current problems in biology is to discover the function of these proteins that are currently unknown, and to better understand the function of those that are putatively known. To help do this biologists require new and powerful tools to browse and compare bioinformatic databases, and so extract the wealth of information hidden in them.

Many bioinformatic databases are publicly available: e.g., the whole genome of the Yeast is accessible from MIPS¹. Also, many tools are available: e.g., PSI-BLAST for comparing sequences, ExPASy for computing physical properties of proteins. However, these data sources and analysis tools are disconnected from each other, making it very difficult to perform genome-wide analysis. Moreover, they usually offer limited forms of querying and navigation.

* This project is funded by the BBSRC grant 21BEP17028.

¹ Munich Information center for Protein Sequences, <http://mips.gsf.de/>

Our aim is to provide biologists with a high-level and integrated interface for browsing and analyzing a whole genome. To do this we first must build a secondary database gathering data from different sources, and represent them in a uniform way. We then must define a querying language that fits the needs of bioinformatics, and allows browsing capabilities. In this paper, we focus on the second task. This language can deal with taxonomies of protein functions, with complex sequence patterns (as in Prosite), and with structures (e.g., the transcription of proteins from several RNA parts called *exons*). The need for complex representations and reasoning mechanisms leads us to the use of logics specialized to bioinformatics. Hence the name of our system, BLID, which stands for Bio-Logical Intelligent Database. The term “intelligent” refers to the automated analysis, such as machine learning or data-mining, that will be made available on top of the querying system in the future.

Section 2 discusses the use of Formal Concept Analysis (FCA) for *bio-logical browsing*, and presents Logical Information Systems (LIS) as a theoretical framework for BLID. Section 3 presents a logic for the representation and reasoning of descriptions and queries. Section 4 explains and illustrates how an automatic and non-hierarchical navigation can be combined with logical querying. Finally, Section 5 discusses related works, and Section 6 draws some future directions, especially w.r.t. analyses.

2 Concept Analysis and Logical Information Systems

Formal Concept Analysis (FCA) is a mathematical theory based on ordered sets and complete lattices [Wil82]. A *context* is a triple $(\mathcal{O}, 2^{\mathcal{A}}, d)$, where \mathcal{O} is a set of *objects*, \mathcal{A} a set of *attributes*, and d is a mapping from objects to their description, i.e. a set of attributes. Then, a Galois connection (ext, int) is defined between sets of objects and sets of attributes. For every set of attributes A , its *extent* $ext(A) = \{o \in \mathcal{O} \mid d(o) \supseteq A\}$ is defined as the set of objects whose description contains A (i.e., the answers of A , when A is seen as a query); and for every set of objects O , its *intent* $int(O) = \bigcap_{o \in O} d(o)$ is defined as the set of attributes shared by all objects in O . Pairs of related extent and intent, such as $(ext(A), int(ext(A)))$ or $(ext(int(O)), int(O))$, are called *concepts*, and form together a complete *lattice of concepts* when ordered by set inclusion on their extent (or equivalently on their intent). Numerous works have shown the usefulness of this concept lattice for information retrieval combining querying and navigation [GMA93,FR03], learning and data-mining [GK00,FR02b].

This applicability of FCA to information retrieval and learning is the basis for our choice of its use as a theoretical foundation. In BLID, objects are the ORFs² of some organism (the Yeast in the rest of this paper). However, simple sets of attributes are not an expressive enough language for object descriptions and queries. For example, a protein sequence can not be made an attribute as it

² ORFs (Open Reading Frames) are segments of DNA in chromosomes supposed to be transcribed and translated into proteins. An ORF coincides with the coding region of a gene when this protein has directly been observed.

is different for each gene. A set of predefined patterns could be used instead (e.g., from Prosite), but information about the gene would be lost, and querying with new patterns would no longer be possible. We wish to preserve all information and to dispose of an open query language.

Logical Concept Analysis (LCA, [FR03]) is an extension of FCA that allows for the replacement of sets of attributes $A \in 2^{\mathcal{A}}$ by formulas $f \in L$ of a logic. The formulas need only be ordered by a *subsumption* relation \sqsubseteq , and this must form a lattice. Logical Information Systems (LIS, [FR03]) are founded on LCA and are characterized by: (a) an object-centered representation; (b) a tight combination of querying and navigation; (c) a logical representation of object descriptions, queries, and navigation links; (d) genericity in the logic for customization.

Section 3 describes the building of a logic that is designed specifically for BLID. This comprises the definition of the language, as well as a few necessary operations: (1) the subsumption \sqsubseteq for ordering formulas according to their specificity/generality³ ($f \sqsubseteq g$ means f is more specific than g), (2) the conjunction \sqcap , and (3) the function *feat* that maps each object description to a set of more general formulas, their *features*. These features play an important role in navigation, which is presented in Section 4. They are generated mainly automatically by the operation *feat*, but can also be introduced at any time manually by users according to their needs.

3 Customized Logic and Querying for Bioinformatics

A logical context is build by creating an object for every ORF of the Yeast's genome. Each object/ORF is described by collecting information from various data sources (see Section 1). For instance, a partial description of the ORF YAL003w, incorporating various data types, is (sequences are shortened):

```
[ name is "YAL003w", nb_atoms = 3138, mol_weight = 22.627e3,
  seq is MAS[..]QKL, struc is c(8)a(12)c(3)[..]b(10)c(5)a(25)c,
  some exon is [1,80], some exon is [447,987],
  'mfc05/04/02: elongation' ].
```

This description combines different concrete domains: text (name), integer (number of atoms), float (molecular weight), two kinds of sequences (over amino-acids and 3D structures), segments (exons). The first sequence (attribute *seq*) is made of amino-acids, and defines the protein expressed by the ORF. In solution the protein folds to form a specific 3D-shape. This shape, the tertiary structure, is still generally unknown, but it is possible to reliably predict an intermediate structure, the secondary structure [OK00]. This latter structure (attribute *struc*) is represented as a sequence composed of 3 kinds of structure element (helices *a*, sheets *b*, and connecting elements called coils *c*), which can have different lengths (given between brackets after each structure element). The exons

³ Notice that the left argument in the subsumption relation is not restricted to be an object description, but can be any query as well as the right argument. This makes it much harder to define such logics, but is necessary for navigation.

are the gene segments, from which an ORF is composed. The last term in the description is an element of the taxonomy of functional classes, found in MIPS. (This taxonomy has been used as target classes in machine learning [KKCD00].)

Let the following expression be a query:

```
('mfc05: PROTEIN SYNTHESIS' or 'mfc06: PROTEIN FATE')
and some exon start >= 2 and nb.atoms in 3000..4000 and
seq match N-{P}-[ST]-{P} and not name ends with "w",
```

where the pattern $N\text{-}\{P\}\text{-}[ST]\text{-}\{P\}$ is the Prosite motif PS00001, which is described as “N-glycosylation site”. While none of the terms of this query appears as such in the description of YAL003w, the latter is still an answer of the query. This means that propositional logic is not expressive enough as a query language w.r.t. above description. One could wonder if propositional logic could be made suitable by adapting the object descriptions. The answer is no, because some data types have an infinite number of patterns (e.g., numerical intervals, sequence motifs); and even for data types where this adaptation is possible (e.g., finite taxonomy of functional classes), this would imply redundancy and potential exponential growth of the descriptions.

Our logical language for descriptions and queries can be understood as a propositional logic, whose atoms are replaced by *logical features* belonging to fragments of predicate logic. In fact, this is equivalent to saying that our logic is a controlled fragment of predicate logic, plus some theory about the considered data types. For instance, the logical feature `seq match N-{P}-[ST]-{P}` can be translated into predicate logic by:

$$\begin{aligned} \forall Orf : \exists Start, P1, P2, A2, P3, A3, P4, A4 : & seq(Orf, Start) \wedge \\ & somesucc(Start, P1) \wedge aa(P1, 'N') \wedge succ(P1, P2) \wedge aa(P2, A2) \wedge \\ & A2 \neq 'P' \wedge succ(P2, P3) \wedge aa(P3, A3) \wedge (A3 = 'S' \vee A3 = 'T') \wedge \\ & succ(P3, P4) \wedge aa(P4, A4) \wedge A4 \neq 'P', \end{aligned}$$

given some theory to define the predicate *somesucc* as the transitive closure of predicate *succ*. It should be clear from this example that a customized logic is preferable to predicate logic as a query language. This is more than mere syntactic sugar because the use of specialized logics enables us to make the computation of subsumption decidable, simpler, and more efficient (remembering that subsumption needs to be applied between queries as well).

Building such a logic from scratch would be a tedious task because of the number of different concrete domains. Moreover, this would make it difficult to extend, or to reuse, parts of existing customized logics. In order to favor modularity and re-usability we apply the principles of *logic functors* [FR02a], which enable us to build complex logics by simple composition of smaller logic components, the *logic functors*. Essentially, a logic functor is a function from logics to logics, where logics are modeled as abstract types encapsulating both representation and reasoning. Most logic functors are reused from previous applications, and a few others, specific to bioinformatics, are created (e.g., for protein sequences and Prosite motifs). Due to limited space, we do not give in this paper

formal definitions for the language, the semantics, and the subsumption of logic functors. For those interested, they are available for a few functors in [FR02a].

Figure 1 shows the way the logic functors are composed as a tree. Each node is a logic functor that is applied to the logics composed from its sub-nodes. At the root of the tree we recognize the propositional parts of the logic. The remainder of the tree describes the logic of features that replace the usual atoms. Features are used to represent both object descriptions and query terms. They are essentially conjunctions of terms taken in concrete domains. The functor **Sum** allows us to easily combine any number of concrete domains, and facilitate extensibility of the logic. Finally, the functor **AIK** (named after the epistemic logic *All I Know*) enables to apply the Closed World Assumption on object descriptions [FR02a].

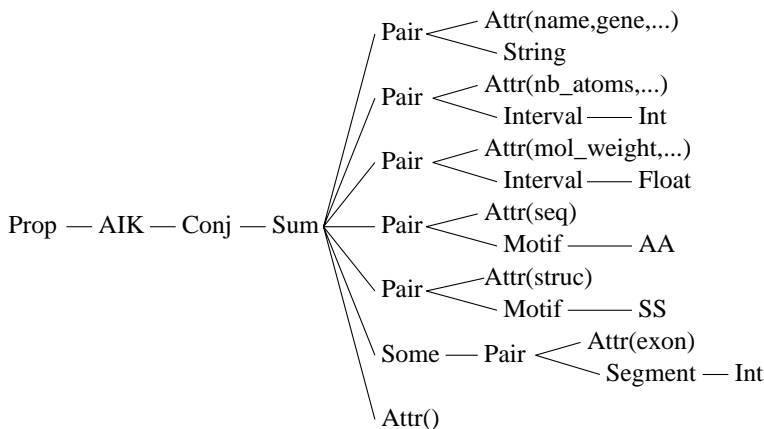


Fig. 1. The BLID's logic represented as a tree of logic functors.

The logical context of chromosome A contains 108 objects (ORFs), from which 4867 features are extracted. This makes an average of 161 features per object, and 3.5 objects per feature (feature sharing). As this context is extended to the whole genome (6141 ORFs), the number of features per object remains constant, and the sharing increases, which results in a total number of features of around 60,000. In such a large context, it becomes intractable to compute the concept lattice. However, it is important to provide users with navigation as they cannot remember by heart the function names or the Prosite motifs, and also because, given some previous query, it is difficult to guess relevant features to refine it. Section 4 develops an interactive and incremental way of building such queries: *logical navigation*.

4 Logical Navigation

The idea of navigation is to help users build their queries, and to enable them to form overviews on the data. In the domain of concept analysis, navigation

is usually realized by a direct browsing of the concept lattice, or some part of it [GMA93]. However, this lattice becomes rapidly very large, and we prefer to realize navigation by a form of human-computer dialog [FR03], as this gives more freedom for controlling the amount of answers.

To navigate from one query/concept to another, users specify query *increments* with *exclamatory commands*, such as `! name ends with "w"`, and they get suggestions for increments with the *interrogative* command `"?"`. These suggestions are found among the features that have been automatically extracted from object descriptions by the logical operation *feat*. For example, in the context made of all ORFs of chromosome A, this command gives the following result:

```
[1] ?                                     What is there ?
100 ! struc                               100 ORFs with known 2nd structure !
101 ! 'MIPS function'                     101 ORFs with function !
108 ? name                               What kind of name ?
108 ? some exon                           What kind of exon ?
108 ? seq                                 What kind of sequence ?
108 ? mol_weight in ..                     What kind of mol. weight ?
108 ? nb_atoms in ..                       What kind of nb. of atoms ?
108 object(s)                             There are 108 selected ORFs.
```

The system returns not only exclamatory suggestions (query increments), but also interrogative suggestions. These can be understood as “questions as answers to questions”, and their purpose is to provide more concise answers, as without them many exclamatory suggestions would possibly replace each interrogative suggestion. These are called *view increments*, because they allow to focus on one kind of features. For instance, the user can select the command `"? nb_atoms in .."` in order to focus on the number of atoms:

```
[2] ? nb_atoms in ..                     What kind of nb. of atoms ?
  5 ! nb_atoms = 2****                    5 ORFs with nb. of atoms in [20000,30000[ !
 22 ! nb_atoms = 1****                    22 ORFs with nb. of atoms in [10000,20000[ !
 81 ! nb_atoms = 0****                    81 ORFs with nb. of atoms in [0,10000[ !
108 object(s)                             There are 108 selected ORFs.
```

The formula `nb_atoms = 2****`, which means the number of atoms is comprised between 20,000 and 29,999, is a feature automatically generated by the functor `Int` to make the navigation more progressive than a flat set of values. This makes the answers look like a histogram, as values at the left of increments are the number of objects they would select (support). With query languages such as SQL or Prolog, one would either get a flat list of all ORFs along with their exact number of atoms, or have to ask an aggregative query for all relevant intervals; which are difficult to know without prior knowledge of the range and the scale of the attribute (which can change according to the working query).

Coming back to command [1], we see that the feature `'MIPS function'` appears as a query increment, because it is not supported by all objects. However, we would expect it to be as well a view increment, focusing on the functions of ORFs. In fact, exclamatory suggestions can often be combined with an interrogative command.

```
[2] !? -l -i 'MIPS function'           Select ORFs with function ! What kind of
                                         functions ?

22 ! 'mfc01: METABOLISM'
3  ! 'mfc02: ENERGY'   -> 'mfc01: METABOLISM'

[../]
39 ! 'mfc99: UNCLASSIFIED PROTEINS'

101 object(s)           There are 101 selected ORFs.
```

The 101 objects are selected and functional classes are listed in lexicographical order (option `-l`). This shows that about 40% of ORFs are unclassified. Option `-i` displays contextual implications between suggested increments. This enables the user to discover that every ORF in chromosome A that has an energetic function, has also a metabolic function.

5 Related Work

Our logics are similar to Description Logics (DL, [Bra79]), in the sense that formulas are variable-free, and their semantics is based on object sets rather than on truth values. Our attributes are equivalent to functional roles, and our operator **some** corresponds to the existential quantification. The two key differences are the modularity of logic functors, and our focus on concrete domains. Furthermore, it would be possible to define a logic functor implementing a description logic in which atoms could be replaced by formulas of concrete domains; as it has been done with propositional logic. Both DL and our logics could be translated into predicate logic, which is more expressive; however they are more readable, and allow for logical navigation thanks to their compatibility with Logical Concept Analysis. We are also developing in parallel a querying interface in predicate logic (using Prolog) to offer more expressive power to expert users, but at the cost that no navigation is provided.

A project related to ours is GIMS [Cor01], which aims at providing querying and analysis facilities over a genome database. In this project, simple queries can be built incrementally by selecting attributes and predefined value patterns in menus. Canned queries are made available for more complex queries and analysis. We differ in that we have made the choice to give users an open language, knowing that navigation will be available to guide users; even if they have no prior knowledge. It is difficult, if not impossible, to forecast all types of queries that may be of interest in the future.

6 Future Work

Our future work concentrates on providing analysis facilities in addition to querying and navigation. The kind of analysis we are mostly interested in is to discover by machine learning techniques rules that predict the biological functions of ORFs from genomic data (i.e., *functional genomics* [KKCD00]). A first step will be to integrate existing machine learning techniques in BLID. Propositional learners (e.g., C4.5, concept analysis [GK00]) expect kind of attribute contexts,

which can easily be extracted from the BLID's logical context by making each feature (e.g., sequence motifs) a Boolean attribute. Inductive Logic Programming (ILP, [MR94]) expects a representation of examples in predicate logic, which can always be obtained by translating them from our specialized logics.

Ultimately, BLID could be made an Inductive Database [dR02] by unifying various machine learning and data-mining techniques under a unified *inductive query language*. For instance, such a language could allow to ask for “all most general rules predicting whether a protein is involved in metabolism according to its sequence”. Such a high-level language would be very helpful to biologists and bioinformaticians, who strive to relate genomic data to biological functions.

A LIS executable for Unix/Linux can be freely downloaded at <http://users.aber.ac.uk/sbf/camelis>.

References

- Bra79. R. J. Brachman. On the epistemological status of semantic nets. In N. V. Findler, editor, *Associative Networks: Representation of Knowledge and Use of Knowledge by Examples*. Academic Press, New York, 1979.
- Cor01. M. Cornell et al. GIMS – a data warehouse for storage and analysis of genome sequence and functional data. In *IEEE Int. Symp. on Bioinformatics and Bioengineering*, pages 15–22. IEEE Press, 2001.
- dR02. L. de Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, December 2002.
- FR02a. S. Ferré and O. Ridoux. A framework for developing embeddable customized logics. In A. Pettorossi, editor, *Int. Work. Logic-based Program Synthesis and Transformation*, LNCS 2372, pages 191–215. Springer, 2002.
- FR02b. S. Ferré and O. Ridoux. The use of associative concepts in the incremental building of a logical context. In G. Angelova U. Priss, D. Corbett, editor, *Int. Conf. Conceptual Structures*, LNCS 2393, pages 299–313. Springer, 2002.
- FR03. S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 2003. To appear.
- GK00. B. Ganter and S. Kuznetsov. Formalizing hypotheses with concepts. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*, LNCS 1867, pages 342–356. Springer, 2000.
- GMA93. R. Godin, R. Missaoui, and A. April. Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. *International Journal of Man-Machine Studies*, 38(5):747–767, 1993.
- Gof97. A. Goffeau et al. The Yeast genome directory. *Nature*, 387:1–105, 1997.
- KKCD00. R. D. King, A. Karwath, A. Clare, and L. Dehaspe. Genome scale prediction of protein functional class from sequence using data mining. In R. Ramakrishnan et al, editor, *ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 384–389. ACM, 2000.
- MR94. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.
- OK00. M. Ouali and R. D. King. Cascaded multiple classifiers for secondary structure prediction. *Prot. Sci.*, 9:1162–1176, 2000.
- Wil82. R. Wille. *Ordered Sets*, chapter Restructuring lattice theory: an approach based on hierarchies of concepts, pages 445–470. Reidel, 1982.

Formal Concept Analysis: Its Role in Teaching Lattice Theory

Brian A. Davey

Mathematics, La Trobe University
Victoria 3086, Australia
B.Davey@latrobe.edu.au

Abstract. In this talk I shall relate some of my experiences in teaching lattice theory and the theory of ordered sets to undergraduates since 1975. I will show how Formal Concept Analysis can be used as a unifying and motivating example in many parts of the theory.

1 The Way It Was

I have taught lattice theory to undergraduates at La Trobe university since 1975. I have also had the opportunity to present one-semester courses at Oxford university and Monash university. Over a period of 13 years, I refined my notes and the exercise sets. Each year, new exam questions were incorporate into the exercises. The wording of exercises, which caused confusion amongst students, were altered and, where necessary, appropriate hints were added. In this way I produced a set of notes which provided an excellent basis for a course pitched at typical second or third year mathematics students. In 1984, Hilary Priestley came to La Trobe to work with me and took a copy of my “*Lattice Theory*” notes back to Oxford with her. She modified them by including more information on Boolean algebras and used them for a course she was teaching to undergraduates in theoretical computer science and mathematics. Meanwhile, I continued to expand my version. In 1987 we proposed to Cambridge University Press that we combine our two sets of notes into a text book. Hilary returned to La Trobe to continue our research in 1988. During this visit we agreed on the overall structure of the text. On her return to Oxford, writing of the first edition of *Introduction to lattices and order* [1] began in earnest.

The structure of the first edition was strongly influenced by the request from our editor, David Tranah, that we make the book as attractive as possible to computer scientists: we wanted to write a text book but he wanted to sell one! We spent a lot of time working through unpublished notes and manuscripts on computer science. As we remarked in the preface, “... *course notes by Dana Scott, Samson Abramsky and Bill Roscoe enticed us into previously unfamiliar territory* ...” David Tranah also asked that the computer science material be as early as possible in the text and as a result, Chapters 3 and 4, on CPOs and Fixpoint theorems, were added.

In order to keep the price within the reach of our students, we set ourselves a strict page limit of 256 pages. Even with the inclusion of the new material in

Chapters 3 and 4, this allowed us a closing 16-page chapter. We made a decision that was considered by some of our colleagues to be rather radical. As our final chapter, we included an introduction to the basics of Formal Concept Analysis, a topic, if not in its infancy, then in its early adolescence. This decision has been more than vindicated over the years.

2 The Way It Is

Between 1990 and 2000, both authors continued to teach subjects based on the text. The chapter on FCA was very soon promoted by both of us (independently) to as early in the semester as possible. Our typical undergraduate courses consisted, in order, of Chapter 1, half of Chapter 2, Chapter 11 (the FCA), followed by Chapters 5 to 8. The remaining chapters (3, 4, 9 and 10) consist of more advanced material taught in the fourth year to students doing an honours degree.

By 2000, the fourth printing of the first edition had sold out and CUP requested a second edition. Based on our experience of teaching from the text between 1990 and 2000, we decided to reorder the chapters of the text and to present them in the order that we taught them. So we find in the second edition of *Introduction to lattices and order* [2], published in 2002, that Formal Concept Analysis has been promoted from Chapter 11 to Chapter 3. This has the distinct advantage that topics introduced earlier, such as join-irreducibility and join-density, see immediate applications in the Fundamental Theorem of Concept Lattices and other topics introduced in later chapters, such as Galois connections and the representation of finite distributive lattices, can be motivated by the results on concept lattices. The chapter on FCA in the second edition also includes a natural algorithm for finding all concepts of a context. While there is software that will do this for us, we believe that it is important that students get their hands dirty with some small examples before handing over responsibility to the technology.

In this talk I will give a number of examples that show how Formal Concept Analysis both reinforces and leads naturally to other important topics within lattice theory.

References

1. Davey, B.A. and Priestley, H.A., *Introduction to Lattices and Order*, Cambridge University Press, Cambridge, 248 + viii pp., 1990.
2. Davey, B.A. and Priestley, H.A., *Introduction to Lattices and Order*, Second Edition, Cambridge University Press, Cambridge, 285 + x pp., 2002.

Concept Lattices for Information Visualization: Can Novices Read Line-Diagrams?

Peter Eklund¹, Jon Ducrou², and Peter Brawn³

¹ School of Information Technology and Computer Science
The University of Wollongong, Northfields Avenue, NSW 2522
peklund@uow.edu.au

² Email Analysis Pty Ltd
PO Box U42, The University of Wollongong
Northfields Ave, NSW 2522, Australia
jon@mail-sleuth.com

³ Access Testing Centre
112 Alexander Street, Crows Nest, NSW 2065, Australia
peterb@testingcentre.com

Abstract. MAIL-SLEUTH is a personal productivity tool that allows individuals to manage email and visualize its contents using line diagrams. Based on earlier work on the Conceptual Email Manager (CEM), a major hypothesis of MAIL-SLEUTH is that novices to Formal Concept Analysis can read a lattice diagram. Since there is no empirical evidence for this in the Formal Concept Analysis literature this paper is a first attempt to test this hypothesis by following a user-centred design and evaluation process. Our results suggest that, with some adjustments, novice users can read line diagrams without specialized background in mathematics or computer science. This paper describes the process and outcomes based on usability testing and explains the evolution of the MAIL-SLEUTH design responding to the evaluation at the Access Testing Centre.

1 Introduction

Mixed initiative [15] is a process in human-computer interaction involving humans and machines sharing tasks best suited to their individual abilities. In short, the computer performs computationally intensive tasks and prompts human-clients to intervene when the machine is unsuited or resource limitations demand human intervention. This process is well-suited to document browsing using Formal Concept Analysis (FCA) and has been demonstrated in previous work in the Conceptual Email Manager (CEM) [5,6,7] and RENTAL-FCA [8]¹.

MAIL-SLEUTH², shown in Fig 1., follows these ideas by re-using the interaction paradigm of the CEM embedded within the Microsoft Outlook email client. Other related work demonstrates mixed initiative using line diagram animation,

¹ <http://www.kvocentral.org/software/rentalfca.html>

² <http://www.mail-sleuth.com>

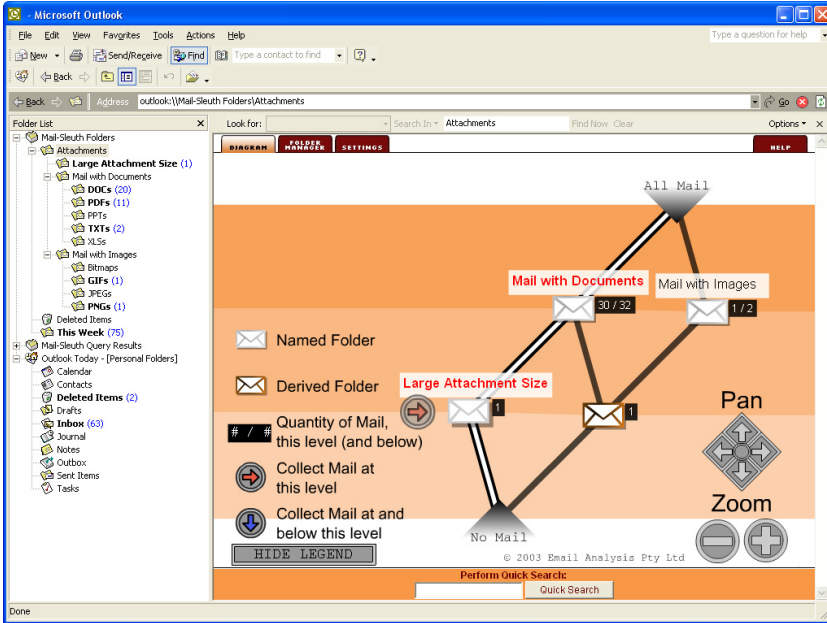


Fig. 1. The final “look” of MAIL-SLEUTH. The line diagram is highly stylized and interactive. Folders “lift” from the view surface and visual clues (red and blue arrows) suggest the queries that can be performed on vertices. Layer colors and other visual features are configurable. Unrealized vertices are not drawn and “Derived” Virtual Folders are differentiated from Named Virtual Folders. A high level of integration with the Folder List to the left and the Folder Manager (see tab) is intended to promote a single-user Conceptual Information System task flow using small diagrams. Nested-line diagrams are not supported, however it is possible to zoom into object sets at vertices with a similar effect.

notably the algorithms in CERNATO [1]³. Like, CERNATO, MAIL-SLEUTH does not employ nested-line diagrams [27,24] instead relying on mixed initiative to reduced line diagram complexity. The client is able to determine trade-offs between attributes and alter search constraints to locate objects that satisfy an information requirement. Because nested-line diagrams are not employed, a major issue is managing the complexity of line diagrams via iterative visualization and zooming. Therefore, keeping the diagram simple needs to be encouraged by the interface. Further, little or no evidence was available in the literature of FCA to support the view that novice individuals could read and interpret line diagrams without specialized training. It was widely assumed that difficulties resulting from novices using a tool like MAIL-SLEUTH would inevitably result. This assumption needed to be firstly tested and secondly, adjustments made in the event that usability problems arose. This paper follows a user-centred test

³ CERNATO is commercial software developed by Navicon AG.

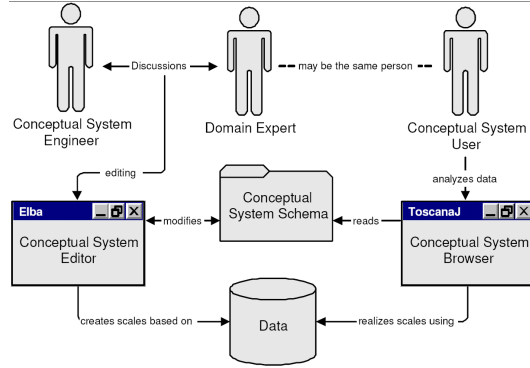


Fig. 2. A Conceptual Information Systems (CIS) and its roles (diagram used with permission of Becker, 2003). Note that a CIS has three roles, the Conceptual Systems Engineering designs scales together with the Domain Expert based on theory or practice. The System User may or may not be the same person as the Domain Expert.

methodology [17], reports its outcomes and the way in which testing conditioned the design of MAIL-SLEUTH and the visualization of line diagrams.

This paper is structured as follows. Section 2 surveys computer-based FCA software systems. A common thread among both commercial and open-source FCA tools is the use of a lattice diagram to visualize information content. MAIL-SLEUTH is situated within this software tools survey. Section 3 covers the background to information landscapes and conceptual knowledge processing. Section 4 describes the evolution of the MAIL-SLEUTH and Section 5 deals with specific evidence that conditioned its design.

2 Tools for FCA

There are two dimensions of software tools using FCA, these are commercial versus open-source and general-purpose versus application specific.

The longest surviving general purpose platform for FCA is the GLAD system [9] which is a general framework for finite lattices, not restricted to FCA. TOSCANA, developed over many years by various members of the Research Group Concept Analysis (*fz^obw*) in Darmstadt, is better known and specifically targeted to FCA. Toscana-systems, referring to outcomes from the TOSCANA software framework, are based on a four-step task flow that includes establishing conceptual scales, data capture, schema browsing and human interpretation. In the usual configuration of Toscana-systems, a program called ANACONDA serves as the conceptual system editor (to define scales), TOSCANA is then the conceptual system browser and data is stored in Microsoft Access. In Toscana-systems there is usually a separation of roles from the individual creating the scales and the end user of the system. The task flow is often called a “conceptual information system” [14], its roles and participants illustrated in Fig. 2.

Modifications to the TOSCANA program have demonstrated that it can be purposed toward specific application problems [10]. In particular, Groh [13] adapted TOSCANA v3.0 to demonstrate the integration of Prediger and Wille's [18] *Relational Power Context Families* in order to represent and process *concept graphs*. However, during this work it became apparent that some of the software libraries on which TOSCANA v3.0 was based, namely embedded graphics libraries from the Borland C++ IDE, would make it difficult for the program to migrate to other operating environments.

In 2000, the GODA project was established as a collaboration between the *Knowledge, Visualization and Ordering Laboratory* (KVO) in Australia and the *fz^obw* in Darmstadt with the vision for a Framework of Conceptual Knowledge Processing. The collaboration produced many outputs, one of which is the TOCKIT⁴ open-source initiative of which TOSCANAJ⁵ forms an integral element. TOSCANAJ is a platform-independent (Java-based) re-implementation of TOSCANA v3.0 that supports nested-line diagrams, zooming and filtering. TOSCANAJ follows the conceptual information systems task flow (shown in Fig. 2) with ANACONDA being replaced by two programs, ELBA and SIENA⁶. ELBA and SIENA are similar with different emphasis – one is a database schema editor the other edits memory-bound schemas. TOSCANAJ can talk to any RDBMS via the ODBC/JDBC or via an embedded RDBMS. Line diagrams of concept lattices can be exported in multiple-formats, color is widely used and TOSCANAJ has more flexible data display features (allowing more varied numerical data analysis and presentations) than TOSCANA v3.0. TOSCANAJ can import legacy file formats from its DOS and Windows-based predecessors, CONIMP [2], TOSCANA and CERNATO, as well as the XML-based conceptual schema format (.CSX).

TOSCANAJ is not the only general multi-platform tool for formal concept analysis to emerge in the open-source era. CONEXP⁷ is another Java-based open-source project that combines context creation and visualization into a single task flow software tool. GALICIA [23] is another Java-based research software program (albeit at an earlier development stage to TOSCANAJ and CONEXP) with particular emphasis on experimentation with lattice closure and visualization algorithms.

Like Groh's adaptation of TOSCANA for concept graphs, TOSCANAJ's source-code has been adapted to various application contexts. Two of these, DOCCO and TUPLEWARE form part of the TOCKIT framework (found at <http://tockit.sf.net>). Tilley [21] has also adapted the TOSCANAJ code in his SPECTRE transformation engine for formal specifications in software engineering.

Prior to 2000, international collaboration in FCA was less organized and open-source software projects less popular than today. WARP-9 FCA [4] was a first attempt at document retrieval based on a faceted hierarchy re-used from a

⁴ <http://tockit.sf.net>

⁵ <http://toscanaj.sf.net>

⁶ The collaboration could not obtain permission to use the name ANACONDAJ.

⁷ <http://conexp.sf.net>

medical ontology and mixed initiative. These ideas were refined and applied to email in CEM [5,6,7] and for the Web in RENTAL-FCA [8] and more recently in the commercial email management program, MAIL-SLEUTH. WARP9, CEM and MAIL-SLEUTH owe their origins to earlier information retrieval tools developed by Carpineto and Romano [3]. Further, this work builds on the idea of FCA for document browsing by Godin and Missoui [12]. Other work that follows this literature thread includes Kim and Compton [16], Rock and Wille [20] and Qian & Feijs [19]. Rapid iteration, direct manipulation to reduce display complexity and the use of conceptual scaling to aid scalability are hallmarks of the later work on document browsing and information retrieval using FCA but there are no existing studies that test the viability of novice users reading and interpreting line diagrams and therefore no indication of the benefit of the work.

3 Information Visualization and FCA

A main attraction of FCA has been its visual utility both for general purpose Conceptual Information Systems frameworks, characterized by Toscana-systems, and also for specialized tools for information retrieval and software engineering. Software engineering has been a strong application area for techniques in FCA and is thoroughly surveyed by Tilley [22,21]. The emphasis on information visualization follows in a natural way from Wille's vision of "landscapes of knowledge" which helps define conceptual knowledge processing.

"The name TOSCANA (= Tools of Concept Analysis) was chosen to indicate that this management system allows us to implement conceptual landscapes of knowledge. In choosing just this name, the main reason was that Tuscany (Italian: Toscana) is viewed as the prototype of a cultural landscape which stimulated many important innovations and discoveries, and is rich in its diversity ..." [26].

Despite the attraction of line diagrams to those of us within the field, it is apparent that the uninitiated have had difficulties interpreting a line diagram as an information space. The conventions for reading line diagrams are manifest in the earliest literature on FCA and these are (in large) related to lattices being drawn on paper (or on a blackboard using chalk). It is difficult from within the field to understand the difficulties faced by novice users or break tradition to develop new conventions for drawing line diagrams. Even the use of color in TOSCANAJ attracts critique from FCA-purists but needs to be situated in the context of a move away from a paper-based approach to Conceptual Information Systems to a more screen-based mixed-initiative interactive approach.

In the context of the design of a commercial tool like MAIL-SLEUTH it is possible to break with tradition and invent (or re-invent) metaphors more suitable to individuals without specialist training in FCA. This process follows a form of user-centered design [17]. The usability tests at Access Testing Centre (ATC)⁸ requirements and condition the software design to make line diagrams more easily understood as an information space by novice users.

⁸ <http://www.testingcentre.com.au>

4 Usability Evaluation

4.1 Comparative Functionality Review

The comparative functionality review was conducted by two ATC analysts performing self-determined exploratory testing. The evaluation had a significant comparative, comparing the ease of executing various key functions in MAIL-SLEUTH against competitor applications.

The May 2003 version of MAIL-SLEUTH had no initial virtual folders when the program was first installed and the user had to go through a folder configuration process from scratch. In FCA terms, MAIL-SLEUTH had no pre-defined conceptual scales. While this may suit advanced users, or users expecting the program to be a general purpose framework for document browsing using FCA, the majority of users are likely to encounter difficulties with this. ATC recommended a number of useful pre-defined Virtual Folders be employed such as “This Week”, and “Attachment” folders for email attachments of different document types and sizes. These form the basis of the Folder List shown to the left of Fig. 1. This recommendation was followed and in subsequent versions pre-defined Virtual Folders were added including the folders mentioned above (various popular document and image attachment types and sizes) and also a “follow-up” folder which tests the Outlook follow-up flag. These serve as examples and a useful starting point from which users can extend the Virtual Folder structure (scale) while benefiting immediately from the software. Other comparable products derive Virtual Folders from reading the mailbox but the structure (once built) cannot be modified or extended as with MAIL-SLEUTH. This advantage is highlighted by including an extensible pre-defined folder structure when the MAIL-SLEUTH program is first installed.

The same time that re-defined Virtual Folders were added, the idea of “User Judgments” (reported in [7]) were eliminated. User Judgments allow the user to over-ride the automatic classification specified in the attached Query of the Virtual Folder. Emails could be drag-and-dropped from regular folders into (or out of) existing Virtual Folders. Access Testing Centre (ATC) found this to be a powerful (and surprising) feature but one that would appeal only to expert users. While the code base for User Judgments still exists under the MAIL-SLEUTH hood it is not presently activated by the interface.

4.2 User-Based Evaluation

The user-based evaluation involved one-on-one interviews and was intended to evaluate the ease of use and expectations of the user community. Six users were drawn from the core target demographic. There was a balance of male and female degree qualified individuals who had expressed an interest in new techniques to categorize and handle their email. Ages spread from 25 to 50 – at least one under 30, at least one over 40. Included in the group were a Librarian, an Insurance Manager, a Financial Analyst, a Recruitment Manager, an Imaging Specialist and a Personal Assistant. Later, informal tests carried out according

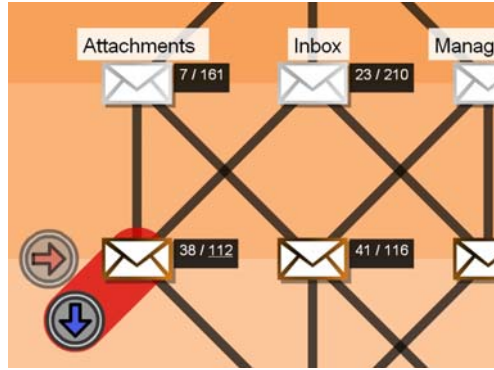


Fig. 3. The red (=extent) (\rightarrow) and blue (=contingent) (\downarrow) “pop up” on roll over of the envelop vertex. The extent and contingent sizes are indicated next to the envelop. The cooresponding white numerals on a black background are underline on rollover with the appropriate arrow.

to the ATC script included a Property Development Manager and a Graduate Software Engineer. Each user session lasted at most 90 minutes and was directed by a usability analyst who observed tasks and recorded relevant data. Each session was then analyzed to identify any usability issues and compile quantitative measures.

4.3 Findings and Actions

The majority of participants were able to learn the basic operations associated with MAIL-SLEUTH and complete a small number of pre-defined tasks. With a simple orientation script (in the place of a help system, incomplete at that point), participants could quickly learn to use the software. For example, once introduced to the concepts of Virtual Folders and how they are associated with a Query (or Queries), participants were able to use the application to create their own folders and populate them with appropriate queries. Participant’s indicated they found the interface reasonably intuitive and easy to use.

“An encouraging finding was that participants were able to read the lattice diagrams without prompting. Subject six even used the word lattice without it having been mentioned to her. Participants correctly interpreted the major elements – for example, how the ‘envelope’ icons related to the mail folders and how derived vertices represented the intersection of two folders”. (ATC Final Report, Usability Analysis, September 2003)

There were still a number of improvements that could be made to the visualization map, in order to present the lattice more clearly:

- *The start and end nodes could be removed from the legend and blue and red arrows could be added.*

The introduction of the red and blue arrows into the lattice diagram is intended to highlight the interactive nature of the lattice diagram as a tool for querying emails. This compensates the interface for the fact that only named folders can be accessed via the Folder List. The red and blue arrows are clues from the line diagram that the *extent* and *contingent* are available and that Derived Folders can be created by manipulating the diagram.

- *For more complicated structures, less emphasis could be placed on regions that are essentially 'unmatched'. This would reduce visual clutter and further highlight the relationships that do exist.*

This comment resulted in the elimination entirely of vertices at unrealized vertices in the line diagram. Many of the test subjects expressed this idea in the usability script. The introduction of the reduced line-diagram was included as an option for advanced users.

- *The format for representing total and dispersed emails associated with each folder could be more clearly represented – some users indicated that the present format (using brackets) represented total and 'unread' e-mails. A reference to the format could be included in the legend.*

Tying together the textual representation of extent and contingent to the red and blue arrows (as shown in Fig. 3) resulted and the total (extent) and dispersed (contingent) sizes being represented as a fraction.

- *The initial/default node view could be improved - when elements are close their labels can overlap. An interesting finding was that some users found more complicated diagrammatic representations better conveyed the relationships to the left-hand folder list.*

The ability to adjust the highlights and font sizes for diagram labels was included (along with the ability to color the layered highlights). The observation that more complex line diagrams more strongly linked the line diagram to the Folder List is because a larger line diagram contains more labels appearing in the Folder List. Thus, the correspondence from line diagram to Folder List is more easily made when there are a larger number of intersecting elements.

Finally, user responses in this small demographic give encouraging indications of an implicit understanding of information visualization using line diagrams. When shown a very large line diagram our librarian found it overwhelming but was certain that there was “value in a lattice of the information space”. More specifically, one user said that she preferred a reduced line diagram, namely she saw “no reason that points without corresponding data should be drawn at all”.

When asked what they liked most about the application users responded with statements such as; “Defined searches - better time management. Ability to separate text from e-mails, program creates folders for you”. We interpret this to mean that this user understands that a permanent standing Query is created attached to a Virtual Folder. The term “Virtual Folder” was also used by another respondent when asked the same question “Drilling down through virtual folders to locate specific emails etc.”, this indicates a familiarity with the idea of a “Virtual Folder”, either pre-existing or learned during the 30-40 minutes using

Table 1. Participants were presented with a number of statements and they were asked to select a rating. The range of the ratings went from -2 to +2, to indicate the extent to which they agreed with the statement. Here -2 = 'Definitely No', 0 = 'Uncertain' and +2 = 'Definitely Yes'.

	Statement	Ave. Resp.
1	Clear how the application is to be used	1.3
2	The interface was simple to use	0.8
3	The application appears to be a useful tool	1.8
4	I liked the layout of the pages	1.2
5	I found the icons intuitive	0.5
6	I found the Quick Search feature was useful	1.0
7	I found the folder view intuitive	1.3
8	I found the diagrammatic view intuitive	0.8
9	Clear relationship, folder view to diagrammatic view	0.7
10	The configuration functionality was useful	0.8
11	I would use this application	1.7
12	I will recommend this application to others	1.7

the program. Further, the use of the term “drilling down” in the appropriate context of data mining and visualization suggest an encouraging level of comfort among the target user group with the terminology of the program.

Table 1 shows that the user group could use MAIL-SLEUTH and had a clear understanding of its utility. While questions 8 & 9, which relate to visualization of line diagrams, scored relatively poorly compared to other questions, it is apparent that the results are nonetheless positive and doubtful that other question groups would have been so highly scored if the line diagrams had not been understood. Nonetheless, improvements to the visualization aspects of the program did result, mostly on the basis of the user’s written comments, and these are described in Section 5.

Considerable time is spent in the development process responding to negative comments by users during software evaluations. Negative comments were solicited when the group were asked “what they liked least” about the MAIL-SLEUTH application. Responses included: “it takes a few moments to understand the 3-D concept as most people are used to a flat & hierarchical folder layout”. The response to this has been to include a careful introductory tutorial/help system to explain Virtual Folders and Structures and introduce specific “simplified” terminology to facilitate an understanding of MAIL-SLEUTH. It is noteworthy that the Virtual Folder idea also appears as one of the features that people liked most. The comment that the “diagram is a bit overwhelming and has badly chosen colors” was addressed by giving people the option of choosing their own color schemes and font sizes and trying to simplify the line diagram as described in the next section.

5 Design Aids for Interpreting Line Diagrams

During the comparative review of MAIL-SLEUTH in May 2003 a comment was made by co-author Peter Brawn that, “the drawing conventions for a lattice

diagram were no different from a graph in Mathematics. What makes this a lattice diagram and not a graph? How do I know that I should read this top to bottom?”

A line diagram (or concept lattice) is a specialized Hasse diagram with several notational extensions. Line diagrams contain vertices and edges with the vertices often labeled dually with the intent (above) and extent (below). Rather than labeling each node in the line diagram with its intent and extent a reduced labeling scheme can be used and each object (and attribute) appears only once. In many Toscana-systems (and in CEM) a listing of the extent is often replaced with a number representing the cardinality of the extent (and/or the contingent).

In Hasse diagrams, edges (representing the cover relation) are unlabeled. It is well understood in Mathematics that an ordered set is transitive, reflexive and antisymmetric. To simplify the drawing of a ordered set (via its cover relation) the reflexive and transitive edges are removed, and the directional arrows of the relation are dropped. It is therefore meant to be “understood” that the Hasse diagram is hierarchical with the edges pointing upward. In other words, if $x < y$ in the poset then x appears at a lower point than y in the diagram.

“The highlighting of adjoining lines is meant to illustrate relationships within the lattice and this could be clearer. There is a hierarchy within the lattice, which could be reinforced through the use of arrows on connecting lines that appear upon rollover..” (ATC Functional Testing Report, May 2003)

Access Testing Centre (ATC) suggested arrowheads be used in the line diagram to reinforce its hierarchical character. This represents an unacceptable violation of a convention dating back to (at least) to Helmut Hasse’s 1926 book *Höhere Algebra*, so some other mechanism to reinforce hierarchy without tampering with the edge notation in the line diagram had to be found.

To insinuate structure the idea of a layered line diagram was introduced. The principle is iterative darkening with dark at the top to light at the bottom, shades progressively lighter as one moves from one level to the next. This is shown in Fig. 4. The top and bottom elements of the lattice have also been replaced with special icons indicating “All Mail” and “No Mail” (when the bottom element is the empty set of objects). In combination, layering and icon shapes are intended to suggest the top-to-bottom reading of the line diagram.

Shading does not interfere with the conventions of drawing line diagrams because it operates as a backdrop to the line diagram. It can also be turned off if the line diagram is to be embedded in a printed document. However, the interaction of the layout algorithm and background layering fails (background layers are not aligned) in line diagrams with high dimensionality as shown in Fig. 5 (left) requiring human intervention to produce something readable as shown in Fig. 5 (right). It is possible to use the alignment of the background layers to guide the manual layout process. Nonetheless, once layering was used, it was apparent from test subjects that they were (without prompting) able to explain (and read) the line diagram from top-to-bottom and bottom-to-top.

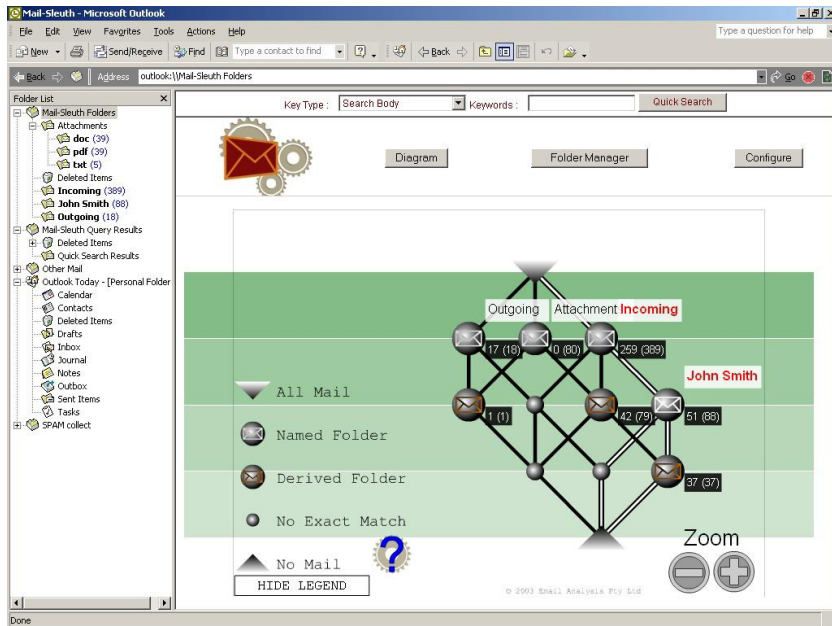


Fig. 4. A line diagram from the August 2003 version of MAIL-SLEUTH. Layering is evident to suggest a hierarchical reading. Top and bottom elements have been especially iconified as arrowheads. Unrealized vertices are differentiated. Realized vertices have been split into two iconic categories “Named Folders” with an intent label with a white envelop and “Derived Folders”, whose intent needs to be ‘derived’ as an orange envelop. Cardinality labels have been replaced with dual labels for “extent (contingent)”. Users complained that the help system was hard to activate or they couldn’t find it and did not recognize the “?” icon as being related to “help”! Note the inclusion of a Quick Search bar at the top which provides an starting point for search.

“It was observed that most nodes in the lattice are depicted using the exact same icon, even though there are a variety of nodes. In particular, the root node, which represents the set of all emails, should be differentiated from all other nodes..” (ATC Report, May 2003)

In MAIL-SLEUTH (and in CEM) the top of the lattice represents all emails in the collection. Some of the vertices shown in the line diagram correspond with actual Virtual Folders that exist in the Folder List to the left, while other vertices represent derivations of the named Virtual Folders. It is useful to indicate, through different icon types, which vertices are named Virtual Folders (appearing in the Folder List), and which are derived. This led to the idea of a “Derived Folder”, a type of Virtual Folder that does not appear in the Folder List and whose name is “derived” from the named Virtual Folders (attribute names) above it in the line diagram.

The number of e-mails represented in each node could also be more clearly illustrated. For example, where totals for vertices and intersections are concerned,

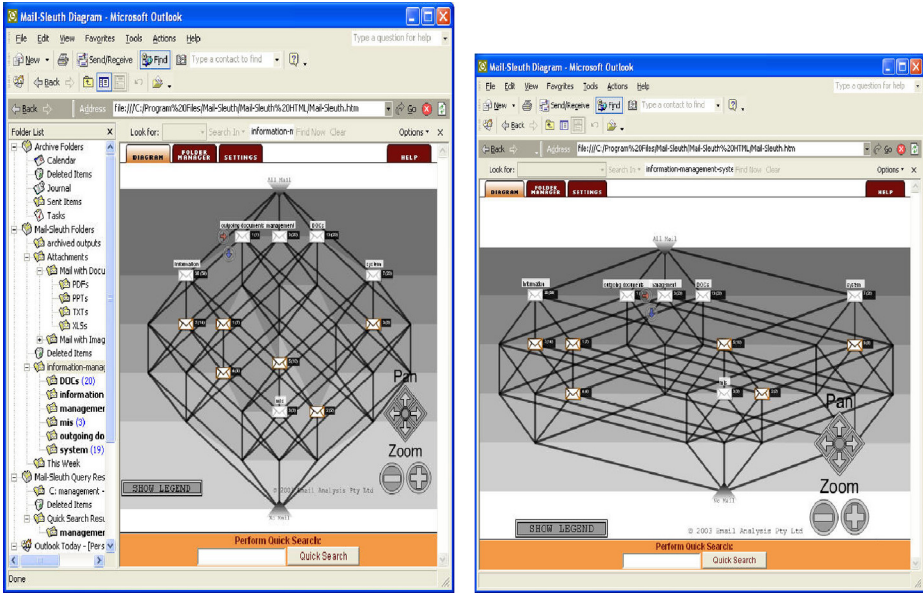


Fig. 5. The interaction of the line diagram layout algorithm and background layering can produce an odd effect (left) but with human intervention layering can also be used as a guide to adjust the line diagram by hand (right) by moving vertices to align the layers. Note that the buttons in Fig. 4 have been replaced with tabs and that the help system is more consistently located to the top right. The Quick Search bar is visually highlighted and placed toward the bottom of the screen for greater emphasis.

two numbers could be displayed corresponding to the extent and contingent size in the form `extent_size (contingent_size)` in Fig. 4.

When drawing “reduced line diagrams” vertices which are unrealized are excluded but automatic layout can be problematic. Because MAIL-SLEUTH was designed for the non-expert, it was decided early to compromise to ensure that the lattice diagram was always “readable” as a default layout and reduced-line diagrams not used as a default. Where elements of a scale are unrealized, the entire label is excluded from the diagram however what remains is drawn as a boolean lattice with an option for a reduced line diagram. This means that certain combinations of realized scale elements may themselves be unrealized. Convention dictated that these be displayed as a vertex in the lattice somehow distinguishable from realized vertices (or not at all). In Fig. 6 unrealized vertices are the same shape and size as realized vertices, the only difference being the presence or otherwise of an envelop icon within the vertex. To distinguish unrealized from realized vertices they were reduced in size as shown in Fig. 4. Top and bottom vertices (when the bottom was an empty set of objects) were also iconified. In addition, realized vertices are identified in two ways. The first where the intent label matches a “Named Folder” in the Folder List of Outlook (to the left of Fig. 1). The second, where vertices representing the intent labels

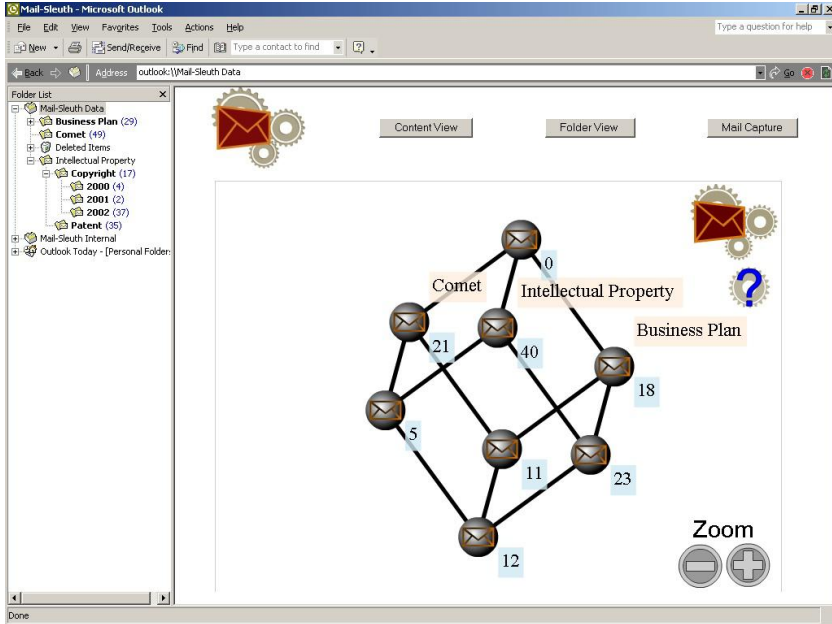


Fig. 6. A line diagram from the May 2003 version of MAIL-SLEUTH. Most of the usual FCA line diagram labeling conventions are followed with the exception of iconifying vertices with an envelop. There is no obvious “search point” (meaning no clear starting place to commence the search) and limited visual highlighting in the diagram itself. Structural diagrammatic constraints are imposed, concepts cannot be moved above their superconcepts for instance.

of the upper covers, these may have common attribute names (Named Folder names) and are colored orange. To avoid cluttering the diagram with labels on all vertices the interface gives scope to query a orange envelop and the result is a new Virtual Folder named after the intent labels of its upper covers appearing in the “Mail-Sleuth search results” in the Folder List.

“.. get rid of the grey blobs...” [User 2]

Because we are dealing with objects that are emails and not ball bearings it was natural to replace the stylized vertices (a legacy of the Hasse diagram) with a literal iconic representation relevant to the domain. In the case where “Derived Folders” were unrealized, no vertex is drawn, where data is present a envelop replaces the envelop/ball icon combination as shown in Fig. 1. Top and (empty) bottom vertices appear at most once in a line diagram and so are removed from the legend (shown in the legend of Fig. 4 but not in Fig. 1) and labeled accordingly in the diagram itself (shown in Fig. 1). The ability to manipulate the line diagram in four directions via the “Pan” widget appears in Fig. 1, and the envelops animate by “appearing to lift” on rollover with drop

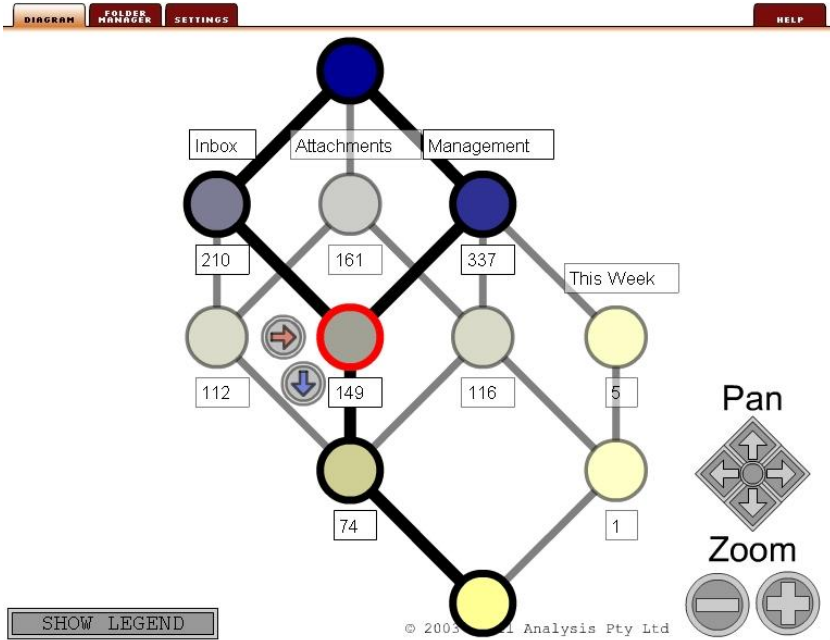


Fig. 7. MAIL-SLEUTH tries to accommodate a new user community to document browsing using FCA. HIERMAIL (shown above) has a much stronger conformity to diagrammatic traditions in FCA. It is effectively a version of MAIL-SLEUTH with a TOSCANAJ-like skin.

shadowing helps suggest that vertices in the line diagram can be moved and therefore manually adjusted by the user.

“Hide the lattice-work where no relationships exists.” [User 6]

Edge highlighting has been used to emphasis relationships in line diagrams in both TOSCANAJ and in CEM. This idea is mainly used as a method to orient the current vertex in the overall line diagram so that relationships can be identified. TOSCANAJ allows the edges of the line diagram to be labeled with the ratio of object counts to approximate the idea of “support” in data mining. That program also uses the size of the extent to determine the color of a vertex. A number of other significant functions for listing, averaging and visualizing the extent at a vertex are also provided by TOSCANAJ.

Trying to create a new user community with MAIL-SLEUTH is an interesting exercise but the original user community also requires attention. HIERMAIL⁹ is a version of MAIL-SLEUTH for the FCA community that conforms to the diagrammatic conventions of TOSCANAJ. It took only a matter of days to rollback the design lessons learned from over four months of usability testing and design refinement with MAIL-SLEUTH to produce HIERMAIL as shown in Fig. 7.

⁹ <http://www.hiermail.com>

6 Conclusion

This paper canvasses a number of usability and visualization issues for interpreting and understanding line diagrams based on the design experience gained from testing MAIL-SLEUTH. It tests, on a very small scale, the ability of novices to FCA to understand line diagrams. The results are promising and indicate that novice users can read and interpret line diagrams.

The design choices made do not represent the only possibilities for helping novice users understand lattice diagrams but rather are determined by constraints on time, resources and programming utilities available to the MAIL-SLEUTH platform. Nonetheless, the choices were suitably tested and result in promising outcomes, users untrained in FCA were able to read and interpret line diagrams and this discovery argues for a less complex task-flow for domain-specific applications such as MAIL-SLEUTH. Naturally, “deep knowledge” can only be gained by complex scale interaction and there is very little of that in MAIL-SLEUTH at its present stage of development. The test candidates were only confronted with small diagrams, direct products of chains, and did not zoom into vertices (although that functionality exists in MAIL-SLEUTH).

The results are therefore preliminary and anecdotal but the methodology followed is a limited example of user-centered designed based on a case-study. Some aspects of the presentation of line diagrams are impossible to adjust but other devices can be introduced to give visual clues on the correct meaning of the diagram and its interactivity. In the process of experimenting with these ideas this paper catalogs the design evolution of the MAIL-SLEUTH program. A much larger usability test needs to be undertaken to verify the findings. These early results are however promising indicators that novice users can read line diagrams.

Acknowledgment

The GoDA project is supported by the Australian Research Council and the *Deutsche Forschungsgemeinschaft*. This research also benefited from the support of an AusIndustry COMET grant. The authors also acknowledge the inputs of Peter Becker, Richard Cole and Thomas Tilley. John Eklund carefully read and commented on the usability aspects of this paper.

References

1. Becker, P., “Multi-dimensional representations of conceptual hierarchies”, in *Conceptual Structures– Extracting and Representing Semantics* contributions to ICCS 2001, pp. 145-158, 2001.
2. Burmeister, P. “Formal Concept analysis with ConImp: Introduction to Basic Features”, TU Darmstadt, Germany, Tech Report, 1996.
<http://www.mathematik.tu-darmstadt.de/~burmeister>

3. Carpineto, C., Romano, G., ULYSSES: A Lattice-based Multiple Interaction Strategy Retrieval Interface. In: Blumenthal, Gornostaev, Unger (eds.): Human-computer interaction: 5th Int. Conf. Proc. of EWHCI '95, Moscow, 1995.
4. Cole, R and P. Eklund, Scalability in Formal Concept Analysis, *Computational Intelligence*, **15** (1), pp. 11-27, 1999.
5. Cole, R and P. Eklund, Analyzing an Email Collection using Formal Concept Analysis, *European Conf. on Knowledge and Data Discovery, PKDD'99*. pp. 309-315, LNAI 1704, 1999. Springer Verlag, 1999.
6. Cole, R. and G. Stumme: CEM: A Conceptual Email Manager *Proceeding of the 8th International Conf. on Conceptual Structures, ICCS'00*, LNAI 1867, pp. 438-452, Springer Verlag, 2000.
7. Cole, R, P. Eklund, and G. Stumme, CEM — A Program for Visualization and Discovery in Email, In D.A. Zighed, J. Komorowski, J. Zytkow (Eds), *Proc. of PKDD'00*, LNAI 1910, pp. 367-374, Springer-Verlag, 2000
8. Cole, R. P. Eklund and F. Amardeilh, Browsing Semi-Structured Texts on the Web Using Formal Concept Analysis, *Web Intelligence*, (eds) Zhang and Springer Verlag, 2003.
9. Duquenne, V., "Lattice structures in data analysis", *Theoretical Computer Science*, **217** (2), pp. 407-436, 1999.
10. Eklund, P., B. Groh, G. Stumme and R. Wille, A Contextual-Logic Extension of TOSCANA, in *International Conference on Conceptual Structures*, pp. 453-467, 2000, CITESEER.NJ.NEC.COM/EKLUND00CONTEXTUALLOGIC.HTML
11. Ganter, B and R. Wille: *Formal Concept Analysis: Mathematical Foundations* Springer Verlag, 1999.
12. Godin, R., R. Missaoui and A. April, Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods, *International Journal of Man-Machine Studies*, 38, 747-767, 1993.
13. Groh, B, A Contextual-Logic Framework Based on Relational Power Context Families, PhD Thesis, Griffith University, 2002.
<http://www.kvocentral.com/kvopapers/grohphd2002.pdf>
14. Hereth, J., G. Stumme, R. Wille and U. Wille: Conceptual Knowledge Discovery and Data Analysis, in *Proceeding of the 8th International Conf. on Conceptual Structures, ICCS'00*, LNAI 1867, pp. 421-437, Springer Verlag, 2000.
15. E. Horvitz, Uncertainty, Action and Interaction: In pursuit of Mixed-initiative Computing *Intelligent Systems IEEE*, pp. 17-20, September, 1999
<http://research.microsoft.com/~horvitz/mixedinit.HTM>
16. Kim, M. and P. Compton, Incremental Development of Domain Specific Document Retrieval Systems, KCAP 2001, Workshop on Knowledge Mark-up and Semantic Annotation, pp. 69-77, 2001.
17. Norman, D.A. and S.W. Draper, *User-centered System Design*, Hillsdale, N.J: Lawrence Erlbaum, 1986.
18. Prediger, S. and R Wille, The Lattice of Concept Graphs of a Relationally Scaled Context, in *International Conference on Conceptual Structures*, pp. 401-414, 1999, CITESEER.NJ.NEC.COM/PREDIGER99LATTICE.HTML
19. Qian, Y and L. Feijs: Step-wise concept lattice navigation, in *Using Conceptual Structures*, contributions to ICCS 2003 (eds) B. Ganter and A. de Moor, pp. 255-267, Shaker Verlag, 2003.
20. Rock, T., Wille, R. Ein TOSCANA-Erkundungssystem zur Literatursuche. In: G.Stumme und R.Wille (Hrsg.): *Begriffliche Wissensverarbeitung. Methoden und Anwendungen*. Springer, Berlin-Heidelberg, 239-253, 2000.

21. Tilley, T., Formal concept analysis applications to requirements engineering and design, PhD Thesis, The University of Queensland, 2003.
22. Tilley, T., R. Cole, P. Becker and P. Eklund, "A Survey of Formal Concept Analysis Support for software engineering activities", in *Proceedings of the First International Conference on Formal Concept Analysis - ICFCA '03*, (eds) G. Stumme and G. Ganter, Springer Verlag, February 2003 (*to appear*).
23. Valchev, P., D. Grosser, C. Roume and M. Hacene, "Galicia: an open platform for lattices", in *Using conceptual Structures*, contributions to ICCS2003, B. Ganter and A. de Moor (eds), Shaker Verlag, pp. 241-254, 2003.
24. Vogt, F. C. Wachter and R. Wille, Data Analysis based on a Conceptual File, *Classification, Data Analysis and Knowledge Organization*, Hans-Hermann Bock, W. Lenski and P. Ihm (Eds), pp. 131-140, Springer Verlag, Berlin, 1991.
25. Vogt, F. and R. Wille, TOSCANA: A Graphical Tool for Analyzing and Exploring Data In: R. Tamassia, I.G. Tollis (Eds) *Graph Drawing '94*, LNCS 894 pp. 226-233, 1995.
26. Wille, R. Conceptual Landscapes of Knowledge: A Pragmatic Paradigm for Knowledge Processing In: W. Gaul, H. Locarek-Junge (Eds) *Classification in the Information Age*, Springer, Heidelberg, 1999.
27. Wille, R. Lattices in data analyse: how to draw them with a computer. In Ivan Rival, editor, *Algorithms and order*. Kluwer, Dordrecht-Boston, 1989, 33-58.

Browsing Search Results via Formal Concept Analysis: Automatic Selection of Attributes

Juan M. Cigarrán, Julio Gonzalo,
Anselmo Peñas, and Felisa Verdejo

Departamento de Lenguajes y Sistemas Informáticos
E.T.S.I. Informática, Universidad Nacional de Educación a Distancia (UNED)
{juanci,julio,anselmo,felisa}@lsi.uned.es
<http://nlp.uned.es>

Abstract. This paper presents the JBraindead Information Retrieval System, which combines a free-text search engine with online Formal Concept Analysis to organize the results of a query. Unlike most applications of Conceptual Clustering to Information Retrieval, JBraindead is not restricted to specific domains, and does not use manually assigned descriptors for documents nor domain specific thesauruses. Given the ranked list of documents from a search, the system dynamically decides which are the most appropriate attributes for the set of documents and generates a conceptual lattice on the fly. This paper focuses on the automatic selection of attributes: first, we propose a number of measures to evaluate the quality of a conceptual lattice for the task, and then we use the proposed measures to compare a number of strategies for the automatic selection of attributes. The results show that conceptual lattices can be very useful to group relevant information in free-text search tasks. The best results are obtained with a weighting formula based on the automatic extraction of terminology for thesaurus building, as compared to an Okapi weighting formula.

1 Motivation

Clustering techniques, which are a classic Information Retrieval (IR) technique, are only now becoming an advanced feature of Web search engines. *Vivisimo* (www.vivisimo.com), for instance, performs a standard web search and then provides a hierarchical clustering of the search results in which natural language expressions label each node. The results of the search “jaguar” with Vivisimo automatically displays a taxonomy of results with nodes such as “Car” (referring to the Jaguar car brand), “Mac OS X” (also known as *Jaguar*), or “animal”, which permit a fast refinement of the search results according to the user needs. If the user, for instance, expands the node “animal”, results are classified in four subclusters, namely “wild life”, “park zoo”, “adventure amazon” and “other topics”. Other examples of clustering techniques in the web include the *Altavista* (www.altavista.com) search engine, which displays a set of suggestions for query refinement that produce a similar clustering effect, and the *Google news service* (news.google.com), where news from hundreds of web servers are automatically grouped into uniform topics.

A common feature of such web services is that clustering is applied to a small set of documents, which come as a result of a query (in search engines) or filtering pro-

file. At this level, clustering proves to be an enabling search technology halfway between browsing (as in web directories, e.g. Yahoo.com or dmoz.org) and querying (as in Google or Altavista). Pure browsing is useful for casual inspection/navigation (i.e., when the information need is vague), and querying is useful when the information need is precise (e.g. I am looking for a certain web site). Probably the majority of web searches lie somewhere between these two kinds of search needs, and hence the benefits of clustering may have a substantial impact on user satisfaction.

A natural question arises: can Formal Concept Analysis (FCA) be applied to browse search results in a free text IR system? FCA is a conceptual clustering technique that has some advantages over standard document clustering algorithms: a) it provides an intensional description of each cluster, which makes groupings more interpretable, and b) cluster organization is a lattice, rather than a hierarchy, facilitating recovery from bad decisions while exploring the hierarchy and, in general, providing a richer and more flexible way of browsing the document space than hierarchical clustering.

The idea of applying FCA only to a small subset of the document space (in our case, the results of a search) eliminates some of the problems associated to the use of FCA in Information Retrieval:

- FCA is computationally more costly than standard clustering, but both can be equally applied to small sets of documents (in the range of 50-500) efficiently enough for online applications.
- Lattices generated by FCA can be big, complex and hence difficult to use for practical browsing purposes. In particular, it can produce unmanageable structures when applied to large document collections and rich sets of indexing terms. Again, this should not be a critical problem when the set of documents is restricted in size and topic by a previous search over the full document collection.

But clustering the results of a free text search is not a straightforward application of FCA. Most Information Retrieval applications of FCA are domain-specific, and rely on thesauruses or (usually hierarchical) sets of keywords which cover the domain and are manually assigned as document descriptors (see section on *Related Work*). Is it viable and useful to apply FCA without such manually built knowledge?

The JBraindead system, which combines free-text searching with FCA on search results, is a prototype Information Retrieval system that serves as a testbed to investigate this research question. In this paper, we focus on the first essential aspect on the application of FCA to free-text searching: *what is the optimal strategy for the automatic selection of document attributes?*

In order to answer this question, we first need to define appropriate evaluation metrics for conceptual lattices in free-text search tasks, and then compare alternative attribute selection strategies with such metrics. Therefore, we will start by defining two different metrics related to the user task of finding relevant information: 1) a *lattice distillation factor* measuring how well the document clusters in the lattice prevent the user from accessing irrelevant documents (compared to the original ranked list returned by the search engine), and 2) a *lattice browsing complexity* measuring how many node descriptions have to be examined to reach all the relevant information. An optimal lattice will have a high distillation factor and a low browsing complexity. With these measures, we will compare two different attribute selection criteria: a standard IR weight (Okapi) measuring the discriminative importance of a term with respect to the collection, and a “terminological weight” measuring the ade-

quacy of a term to represent the content of a retrieved subset as compared to the full collection being searched. We will also study which is the adequate number of attributes to build an optimal conceptual lattice for this kind of task.

The paper is organized as follows: Section 2 provides a brief description of the functionality and architecture of the JBraindead system. Section 3 summarizes the experimental setup and the results obtained; Section 4 reviews related work, and Section 5 offers some conclusions and discusses future work.

2 The JBraindead Information Retrieval and Clustering System

JBraindead is a prototype IR system that applies Formal Concept Analysis to organize and cluster the documents retrieved by a user query:

1. Free-text documents and queries are indexed and compared in a vector space model, using standard *tf*idf* weights. For a given query, a ranked list of documents is retrieved using this model.
2. The first n documents in the ranked list are examined to extract, from the terms in the documents, a set of k optimal descriptors according to some relevance weighting formula.
3. Formal Concept Analysis is applied to the set of documents (as formal objects), where the formal attributes of each document are the subset of the k descriptors which are contained in its text.
4. Besides the intensional characterization of each concept node, an additional description is built with the most salient phrasal expressions including one or more query terms. This additional characterization is intended to enhance node descriptions for the query-oriented browsing task that conceptual lattices play in JBraindead.
5. The resulting annotated lattice is presented to the user, which can browse the top n results by traversing the lattice and/or refine the query at some point. In its current implementation, query refinement can only be made as a direct query reformulation.

The core of the process lies in steps 2 and 4. Step 2 determines the attribute set for a given document set, and then, implicitly, also defines the conceptual lattice. Step 4 enriches the intensional description of concept nodes with query-related phrases, defining how the lattice nodes will be presented to the user.

Figure 1 shows the JBraindead interface for the results of the query “*pesticidas en alimentos para bebés*” (pesticides in baby food) when searching the standard CLEF collection of news in Spanish (see next section for details). Both “*pesticidas*”, “*alimentos*” and “*bebé*” appear as second-level nodes in the conceptual lattice. Other attributes automatically selected by JBraindead are “*potitos*” (a kind of baby food which happened to be recipient of pesticides), “*lindano*” (the kind of toxic waste found in the baby food), “*Hero*” (a baby food brand), or “*toxicos*” (toxic). JBraindead also extracts complex node descriptions including node attributes, such as “*alimentos para bebés*” (baby food) or “*agricultura y alimentación*” (food and agriculture).

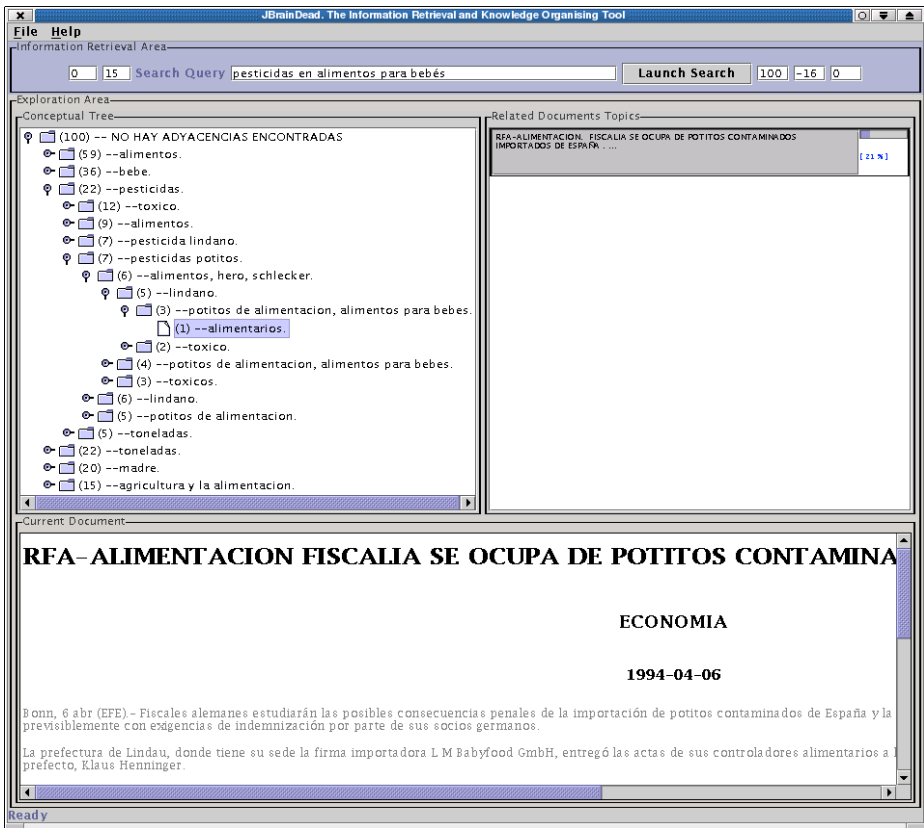


Fig. 1. JBraindead system results for the query “pesticidas en alimentos para bebés” (pesticides in baby food) and the CLEF EFE 1994 news collection.

3 Experiments in Attribute Selection

This section describes a set of experiments designed to find automatically optimal attributes for the set of documents retrieved for a given query in the JBraindead system. We describe a) the Information Retrieval testbed used in our experiments, b) a set of measures proposed to evaluate the quality of conceptual lattices for the purposes of grouping free-text search results, c) the experiments carried out, and, finally, we discuss the results obtained.

3.1 Information Retrieval Testbed

A manual, qualitative inspection of the lattices generated by JBraindead on the results of random queries can provide an initial feedback on the quality of the process. But for a systematic comparison of approaches, an objective measure is needed. While the final purpose of the system is to improve user searches, studies involving users are

costly and should only be performed for final testing of already optimized alternatives; hence, we wanted to find an initial experimental setup in which we could tune the process of selecting document attributes before performing user studies.

The Spanish CLEF EFE-1994 collection that we have used during system development includes a set of 160 TREC-like topics (used in CLEF 2001, 2002 and 2003 evaluation campaigns) with manual relevance assessments from a rich and stable document pool [13], forming a reliable and stable test bed for document retrieval systems. Out of this set, we have used topics 41-87 coming from the CLEF 2001 and 2002 campaigns.

If we feed JBraindead with CLEF topics, we can study how the conceptual lattices group relevant and non relevant documents. The baseline is the ranked list of documents retrieved by the initial search: to discover all relevant documents in the set, the user would have to scan at least all documents until the last relevant document is identified. If the FCA process group relevant documents and the node descriptions are useful indicators of content, then browsing the lattice for relevant documents could provide the same recall while scanning only a fraction of the initial set of retrieved documents, saving time to the user and offering a structured view of the different subtopics among relevant documents.

3.2 Evaluation Measures

How can we measure quantitatively the ability of the FCA process to group relevant documents together? A couple of standard clustering measures are *purity* and *inverse purity*. Given a manual classification of the documents into a set of labels, the precision of each cluster P with respect to a label partition L (containing all documents assigned to the label), the *precision* of P is the fraction of documents in P which belong to L . The *purity* of the clustering is then defined as the (weighted) average of the maximal precision values of each cluster P , and the *inverse purity* is defined as the weighted average of the maximal precision values of each partition L over the clusters. Purity achieves a maximal value of 1 when every cluster has one single document, and inverse purity achieves a maximal value of 1 when there is only one single cluster.

Purity and inverse purity are, then, inadequate measures for the conceptual clustering generated by FCA: the cluster structure of a conceptual lattice is much richer than a plain set of labels; and, in addition, the only distinction that we can make for this experiment is between relevant and non relevant documents. What we want to measure is whether the lattice structure effectively “distillates” relevant documents together, allowing the user to locate relevant information better and faster than in a ranked list of documents. Hence we introduce here a “*lattice distillation factor*” measure which relies on a notion of *minimal browsing area* that we introduce now.

3.2.1 Lattice Distillation Factor

Let C be the set of nodes in a conceptual lattice, where documents are all marked as relevant or non-relevant for a given query. Let us assume that, when visiting a node, the user sees the documents for which the node is their object concept. We will use the term *relevant concept* to denote object concepts generated by, at least, one rele-

vant document, and *irrelevant concept* to denote object concepts generated only by one or more irrelevant documents.

We define $C_{REL} \subseteq C$ as the subset of relevant concepts in the lattice. In order to find all relevant documents displayed in the lattice, the user has to examine, at least, the contents of all concepts in C_{REL} . We define the **minimal browsing area (MBA)** as the minimal part of the lattice that a user should explore, starting from the top node, to reach all the relevant concepts of C_{REL} , minimizing the number of irrelevant documents that have to be inspected to obtain all the relevant information. We can think of the precision of the MBA (ratio between relevant documents and overall number of documents in the MBA) as an upper bound on the capacity of the lattice to “distillate” relevant information from the search results. The lower bound is the precision of the original list: the user has to scan all documents retrieved to be sure that no relevant document is being missed from that list.

The **lattice distillation factor (LDF)** can then be defined as the potential precision gain between the lattice and the ranked list, i.e., as the percentual precision gain between the minimal browsing area and the original ranked list:

$$LDF(C) = \frac{\text{Precision}_{MBA} - \text{Precision}_{RL}}{\text{Precision}_{RL}} \cdot 100 \quad (1)$$

Note that the minimal browsing area and the distillation factor can be equally applied to hierarchical clusters or any other graph grouping search results.

The only difficulty to calculate the distillation factor lies in how to find the minimal browsing area for a given lattice. In order to calculate this area, we will create an associated graph where all nodes are relevant concepts, and where the cost associated to each arc is related to the number of irrelevant documents which will be accessed when traversing the arc. Then we will calculate the minimal span tree for such graph, which will give the minimal browsing area:

1. We start with the original lattice (or any directed acyclic graph). We define the *cost* of any arc reaching a relevant or irrelevant concept node, from one of its upper neighbors, as the number of irrelevant documents that are fully characterized by the node. E.g., if we have an object concept c , such as, $\gamma d_1 \equiv \gamma d_2 \equiv \gamma d_3 \equiv c$, where d_1 and d_2 are non-relevant documents, all arcs reaching c will have a cost of 2.
2. In a top-down iterative process, we will suppress all nodes which are not relevant concepts. In each iteration, we select the node j which is closest to the top and is not a relevant concept. j is deleted and, to keep connections between ancestors and descendants of the node, we create a new arc for every pair of nodes $(u, l) \in Uj \times Lj$, where Uj and Lj are the sets of upper and lower neighbors of j . A cost of $cost(u, l) = cost(u, j) + cost(j, l)$ is then assigned to the new arc. If we end up with more than one arc for a single pair of nodes (u, l) , we select the arc with the lowest cost and suppress the others.
3. The result of the iteration above is a directed acyclic graph whose nodes are all relevant concepts. The minimal span tree of this new graph tells us which is the minimal browsing area in the original lattice.

Figure 3 shows an example of how to build the minimal browsing area and calculate the lattice distillation factor.

3.2.2 Lattice Browsing Complexity

The *distillation factor* is only concerned with the cost of reading documents. But browsing a conceptual structure has the additional cost (as compared to a ranked list of documents) of examining node descriptions and deciding whether each node is worth exploring. For instance, a lattice may lead us to ten relevant documents and save us from reading another ten irrelevant ones... but force us to traverse a thousand nodes to find the relevant information! Therefore, the number of nodes in the lattice has to be considered to measure its adequacy for searching purposes.

There might be also the case that a lattice has a high distillation factor but a poor clustering, forcing the user to consider most of the nodes in the structure. An example can be seen in Figure 2, where all the object concepts occur near the lattice bottom. Precision for the minimal browsing area is 1, and the lattice distillation factor is 100%. The clustering, however, is not good: the user has to consider (if not explore) all node descriptions to decide where the relevant information is located.

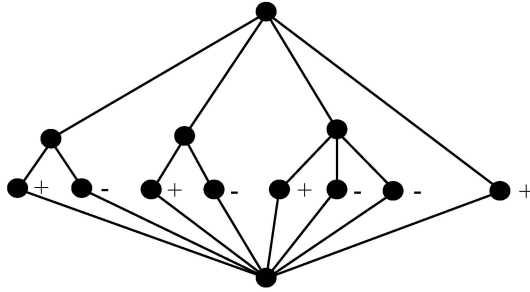


Fig. 2. This lattice has a high distillation factor (LDF = 100%), but the clustering is poor.

We need to introduce, then, another measure estimating the percentage of nodes that must be considered (rather than visited) in a lattice in order to reach all relevant information. We propose a measure of the *lattice browsing complexity* (*LBC*) as the proportion of nodes in the lattice that the user sees when traversing the minimal browsing area. The idea is that, when a node is explored, all its lower neighbors have to be considered, while only some of them will be in turn explored.

Being C the set of nodes in the concept lattice, the set of viewed nodes C_{VIEW} is formed by the lower neighbors of each node belonging to the minimal browsing area. The lattice browsing complexity is the percentage of lattice nodes that belong to C_{VIEW} : $LBC(C) = |C_{VIEW}|/|C|*100$. Figure 4 shows an example of how the lattice browsing complexity is calculated.

3.3 Experiments

We have used CLEF topics 41-87, corresponding to the CLEF 2001 campaign. For each experiment, all topics (title+description) are searched. For every search, a formal context $K=(G,M,I)$ is build, where G is the set of the first 100 documents returned by the search engine in response to a query, M is the set of attributes (variable between experiments), and $d I t$ iff the attribute t is a term occurring in document d .

The two weighting measures used to generate the formal contexts are the Okapi weighting scheme and the terminological formula proposed in [12].

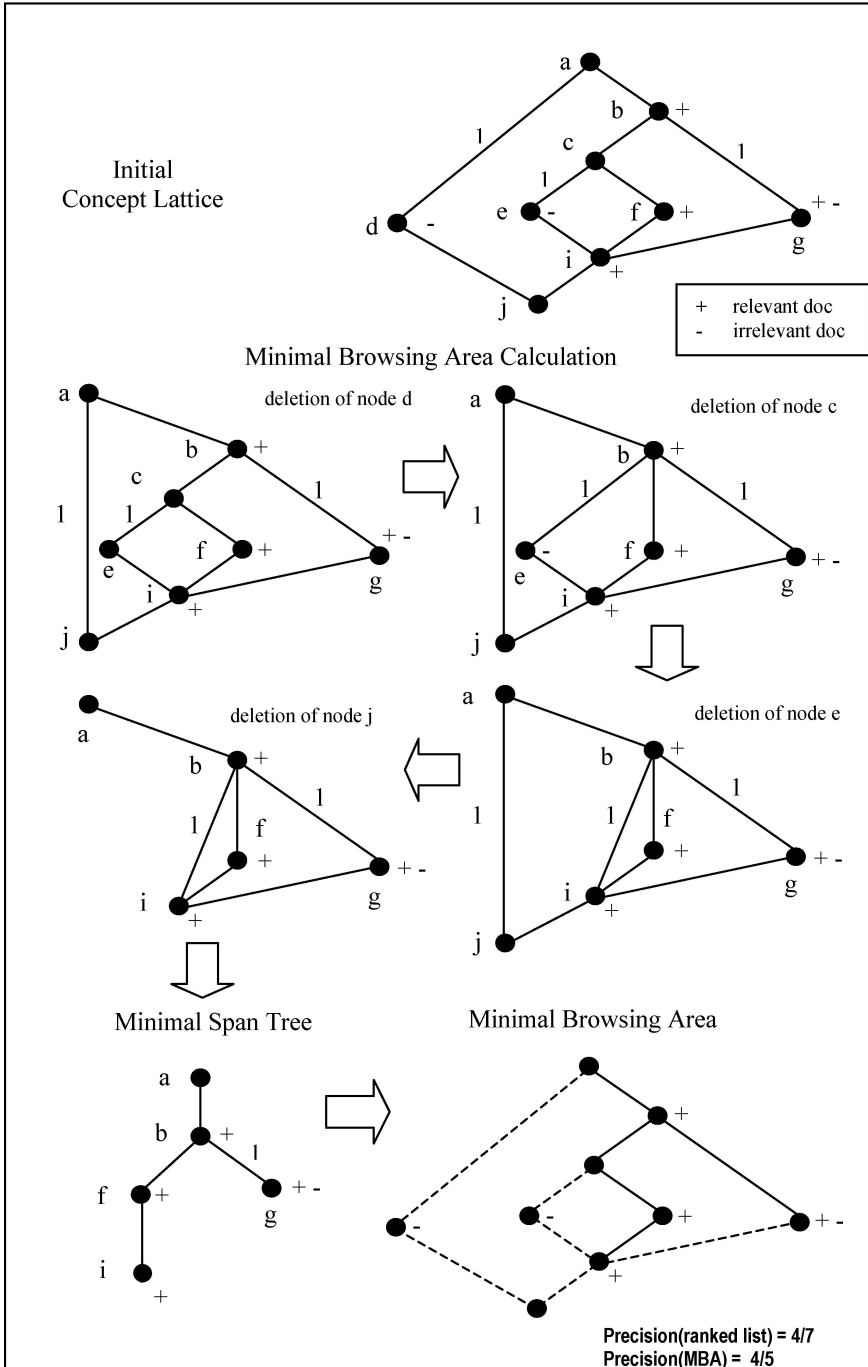


Fig. 3. Calculation of the Lattice Distillation Factor.

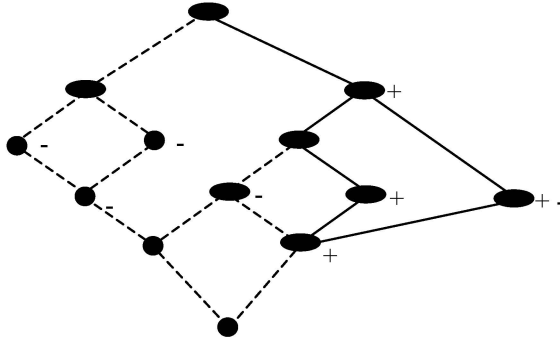


Fig. 4. Concept Lattice with $LBC = 61\%$. *MBA* links are represented as continuous lines on the concept lattice. *LBC* nodes are drawn as oval nodes. Circular nodes represent nodes which are not seen when traversing the *MBA*.

3.3.1 Okapi

Term weights in an IR system measure the importance of a term as a discriminative descriptor for a given document. We have selected the Okapi BM25 weight, which has given the best results for the CLEF collection used in our experiments [15]:

$$w_i = \frac{(k_1 + 1) \cdot tf_i}{K + tf_i} \cdot \frac{\ln\left(\frac{N + 0.5}{f_i}\right)}{\ln(N + 1)} \quad (2)$$

$$K = k_1 \cdot \left[(1 - b) + b \cdot \frac{l_{ret}}{avdl} \right] \quad (3)$$

where the parameters k_1 , b and $avdl$ are adjusted for the Spanish CLEF test collection with the values $b = 0.5$, $k_1 = 1.2$, and $avdl = 300$ taken from [15]. tf_i represents the frequency of the term i in the document (in our case, in the set of retrieved documents), and f_i the document frequency of term i in the whole collection. Finally, l_{ret} represents the total length (in terms) of the retrieved document set.

3.3.2 Terminological Weight Formula

A terminological weight is designed to find, in a collection which is representative from some specific domain, which terms are more suitable as descriptors for a thesaurus of the domain. We use a variation of a formula introduced in [12] which compares the domain-specific collection with a collection from a different domain, and assigns a higher weight to terms that are more frequent in the domain-specific collection than in the contrastive collection. In our case, the domain-specific collection can be the set of retrieved documents, and the contrastive collection is the whole document collection minus the retrieved set:

$$w_i = 1 - \frac{1}{\log_2 \left(2 + \frac{tf_{i,ret} \cdot f_{i,ret} - 1}{tf_{i,col} + 1} \right)} \quad (4)$$

where w_i is the terminological weight of term i , $tf_{i,ret}$ is the relative frequency of term i in the retrieved document set, $f_{i,ret}$ is the retrieved set document frequency of term i , and $tf_{i,col}$ is the relative frequency of term i in the whole collection minus the retrieved set.

3.3.3 Number of Attributes

Finally, for the formulas above we have studied how the number of attributes affects the size and quality of the conceptual lattice. After some initial testing, we have kept the number of attributes between 10 and 20: with less attributes, the clustering capacity is too low, and with more than 20 attributes, the number of nodes becomes exceedingly high for browsing purposes, and the computational cost makes online calculation too slow.

3.4 Results

Table 1 shows the basic outcome of our experiments. In all cases, the Distillation Factor is high, ranging from 346% to 594%. Note that this measure is an upper bound on the behavior of real users: only an optimal traversing of the lattice will give such relative precision gains. Note also that the microaveraged LDF is much higher than would result from the average precisions of the ranked list and minimal browsing area. This is because the LDF expresses the relative precision gain rather than the absolute precision gain.

For 10 attribute terms, the Okapi formula gives a higher Distillation Factor (580% versus 346%) but at the cost of a much larger lattice (70 nodes in average versus 35 nodes with the terminological formula). Both differences are statistically significant according to a paired t-test ($p < 0.05$). In practice, the Okapi formula generates too large lattices for browsing a hundred documents, hence the terminological formula should give better results with experiments involving users.

The LDF seems to grow linearly with the number of attributes, and the complexity factor seems to decay linearly the number of attributes. The number of nodes, however, grows almost exponentially. For 15 terms, the number of nodes generated by the terminological formula is already too large (94 nodes) for practical purposes.

Overall, it seems clear that conceptual lattices can be very effective to group relevant information, and the grouping effect is higher for larger attribute spaces. But the number of nodes quickly becomes impractical. From this point of view, understating attributes as potential terminological units seems to give more compact lattices than seeing attributes as IR indexing terms.

Table 1. Experimental results. The average precision of the original ranked lists was 0.17.

	# Terms	Prec. MBA	LDF	# Nodes	# Nodes Viewed MBA	LBC
Terminological Formula	10	0.35	346 %	35	19	54 %
	15	0.43	493 %	94	50	43 %
	20	0.52	594 %	184	65	36 %
Okapi	10	0.43	580 %	70	32	44 %

4 Related Work

The application of FCAs to Information Retrieval is an increasingly successful field, which has already produced some commercial applications, although all research known to us concentrates on manually (or semi automatically) indexed or classified according to some domain specific thesaurus or classification scheme.

Two early applications for which empirical tests with users were conducted are [11] and [1]. In [11], navigation in a Galois lattice is compared to boolean retrieval and hierarchical navigation in an Information Retrieval task involving users. In the experiment, recall obtained using lattices and boolean retrieval is superior to navigation in a hierarchy. The document collection consisted on 113 short animation film descriptions, and every document was manually indexed by an average of 6.53 classification terms. In [1], a lattice conceptual clustering system is proposed that incorporates background knowledge from the indexing thesaurus (i.e. the broader/narrower term relationships in the thesaurus) into the process of building the conceptual clustering lattice. Browsing with and without the background knowledge were compared in the context of users searchers against a collection of 1555 documents about Artificial Intelligence extracted from a computer engineering collection (INSPEC). Browsing with background knowledge led a 30% relative improvement in recall, showing that the incorporation of specificity relations between indexing terms is a significant improvement over building the lattice without considering the relations between the keywords manually assigned to documents.

One of the application domains that has received more attention is medical documentation. In [3], a set of 9000 patient medical discharge summaries are indexed using SNOMED (Systematized nomenclature of medicine), showing the viability of the approach. The approach has a continuation in [4,2] and [5], where documents are automatically indexed using UMLS (Unified Medical Language System) metathesaurus terms, and the notions of *conceptual scales* and *purified contexts* are introduced for improved, scalable knowledge visualization. Unfortunately, no empirical, quantitative evaluations or user studies have been conducted in this domain, to our knowledge.

FCA has also been applied to document retrieval in conjunction with *faceted thesauruses*, a notion which is related to conceptual scales, in which different aspects of an article description (for instance, the topic of an article and the level of difficulty) have descriptors in different facets of the thesaurus. The IR system *FaIR* [14] is an example of such a system, which is applied to an on-line collection of about 5000 FAQ documents of computing questions. Another application in the computer domain is *Aran* [9], an Information Help System that applies FCA to Unix man pages. A characteristic feature of this system is that it does not employ any prior thesaurus; indexes are obtained from free text in the short (one or two lines) command descriptions that summarize every unix command. As in the medical domain, none of these systems have been quantitatively evaluated.

An attractive example of the possibilities of FCA for knowledge management is the *HIERMAIL* system [8,6], which provides a structured ontology and IR system for Email search and discovery, in which the principles of FCA are supported by an inverted file index that provides efficient client iteration. Although there is no empirical evaluation of the utility of the system (perhaps because it is not trivial to design an evaluation for knowledge management tasks), an indirect evidence of its value is that

the idea of applying FCA to e-mail management has already reached the market with the Mail-Sleuth application (<http://mail-sleuth.com>).

More recently, [7] combine Information Extraction on web documents with FCAs in an information access application on the domain of classified advertisements for Real Estate properties. Rather than simply extracting keywords from documents, the Information Extraction process extracts template-based data to describe advertisements, improving the input to the FCA process.

Finally, a work which is similar in spirit to the JBraindead approach is described in [10]. The authors cluster a news collection (Reuters-21578) combining a standard clustering technique, which is applied to the whole collection, with FCA, which is applied individually to every cluster produced in the first process. One of the salient features of the system is that they use a general purpose lexical knowledge base (WordNet) rather than a domain specific thesaurus as background knowledge, both for the initial clustering process and for the subsequent Conceptual Clustering step. In practice, that means that the input for FCA is closest to free indexing terms than in any of the applications mentioned above. JBraindead uses a similar approach, but in an IR application: Hotho and Stumme apply FCA to smaller subsets of the collection by applying a standard clustering technique, and then performing FCA on every cluster returned; JBraindead applies FCA to smaller subsets of the collection by applying standard IR, and then performing FCA online on the results of the search. It is worth mentioning that in Hotho and Stumme's work, the indexes for the FCA process are the terms with higher values in the centroid vector representing the cluster. The combination of WordNet and centroid vectors is an interesting alternative to the methods evaluated in this paper, which we seek to adapt and compare with our current keyword extraction procedures.

5 Conclusions and Future Work

We have described the JBraindead Information Retrieval system, which combines standard IR techniques with online conceptual clustering applied on the results of the initial user query. The system is domain independent and operates without resorting to thesauruses or other predefined sets of indexing terms. Hence, the contributions of JBraindead to the application of FCA in Information Retrieval lies in the approaches to extract indexing terms for the FCA process and to build natural descriptions of the nodes in the resulting lattice.

In this paper we have focused on the process of attributes selection, comparing two weighting schemas of different nature: the Okapi probabilistic weights, related to the discriminative power of a term for IR purposes, and a terminological weight related to the adequacy of a term as topic-specific descriptor. We have also measured the influence of the number of attributes in the quality of the outcoming lattice for searching purposes. We have made a special emphasis in the definition of metrics to compare different conceptual structures for the task of browsing free-text results, introducing: a) a *lattice distillation factor*, related to how well the conceptual structure prevents the user from reading irrelevant documents, and b) a *lattice browsing complexity*, related to the proportion of nodes in the structure that have to be considered to reach all relevant information. An optimal lattice will have a high distillation factor, a low browsing complexity and a low number of nodes.

The results show that the terminological weighting is better than the IR Okapi weight, and that an increasing number of attributes improves the distillation factor at the cost of a higher browsing complexity. Most differences between runs are statistically significant, showing that the quality of the conceptual structures is highly sensitive to parameter settings.

The JBraindead system illustrates the scalability of FCA to unrestricted Information Retrieval settings, if it is applied to organize search results, rather than trying to structure the whole document collection with conceptual analysis. To our knowledge, this is the first IR system based on FCA that operates on a collection of more than 500 Mb comprising more than 200,000 documents.

JBraindead provides, as well, a test bed to study optimal querying, indexing, visualization and refinement strategies for free-text retrieval based on conceptual clustering. The experiments reported here are just a first step towards optimal, interactive content retrieval and browsing. We are currently experimenting with shallow Information Extraction techniques (named entity recognition, noun phrase indexing) to reach a selection of terms that can be used both to produce better lattice structures and as natural descriptors of nodes.

References

1. Carpineto, C. and Romano, G. A lattice Conceptual Clustering System and Its Application to Browsing Retrieval. *Machine Learning* (1996) 24, 95-122.
2. Cole, R. J. The management and visualization of document collections using Formal Concept Analysis (2000). Ph. D. Thesis, Griffith University.
3. Cole, R. J. and Eklund, P. W. Application of Formal Concept Analysis to Information Retrieval using a Hierarchically structured thesaurus.
4. Cole, R. J. and Eklund, P. W. A Knowledge Representation for Information Filtering Using Formal Concept Analysis. *Linköping Electronic Articles in Computer and Information Science* (2000), 5 (5).
5. Cole, R. J. and Eklund, P. W. Scalability in Formal Concept Analysis. *Computational Intelligence* (1999), 15 (1), pp. 11-27
6. Cole, R. J., Eklund, P. and Stumme, G. Document Retrieval for Email Search and Discovery using Formal Concept Analysis. *Applied Artificial Intelligence* (2003), 17 (3)
7. Cole, R., Eklund, P. and Amardeilh, F. Browsing Semi-structured Texts on the web using Formal Concept Analysis. *Web Intelligence* (2003).
8. Eklund, P. and Cole, R. Structured Ontology and IR for Email Search and Discovery. In *Proceedings of the Sixth Australasian Document Computing Symposium* (2001), Coffs Harbour, Australia.
9. Fernández-Manjón, B., Cigarrán, J., Navarro, A. and Fernández-Valmayor, A. Applying Formal Concept Analysis to Domain Modeling in an Intelligent Help System. In *Proceedings of Information Technology and Knowledge Systems* (1998), 5th IFIP World Computer Congress, Vienna-Budapest.
10. Hotho, A. and Stumme, G. Conceptual Clustering of Text Clusters. In *Proceedings of the FGML Workshop* (2002), Hannover.
11. Godin, R., Missaoui, R. and April, A. Experimental Comparison of navigation in a Galois lattice with conventional Information Retrieval methods. *Int. J. Man-Machine Studies* (1993) 38, 747-767.
12. Peñas, A., Verdejo, F. and Gonzalo, J. Corpus-Based Terminology Extraction applied to Information Access. In *Proceedings of Corpus Linguistics 2001* (2001), Lancaster University.

13. Peters, C., Braschler, M., Gonzalo, J. and Kluck, M (eds.) Evaluation of Cross-Language Information Retrieval Systems (2002). Springer-Verlag LNCS 2406, Berlin.
14. Priss, U. Lattice-based Information Retrieval. Knowledge Organization (2000), 27 (3), p. 132-142.
15. Savoy, J. Report on CLEF 2002 experiments: Combining multiple sources of evidence. In Peters, C., Braschler, M., Gonzalo, J. and Kluck, M. (eds): Advances in Cross-Language Evaluation Retrieval (2003). Springer-Verlag LNCS 2875, Berlin.

Formal Concept Analysis and Semantic File Systems

Ben Martin

Information Technology and Electrical Engineering
The University of Queensland
St. Lucia QLD 4072, Australia
`monkeyiq@users.sourceforge.net`

Abstract. This document presents and contrasts current efforts at applying Formal Concept Analysis (FCA) to some semi structured document collections and file systems in general. Existing efforts are then contrasted with ongoing efforts using the libferris Virtual File System (VFS) as a base for FCA on file systems.

1 Introduction

The file system has become the defacto standard for the storage and management of semi structured data on computers. File systems have evolved from presenting a list of named objects (files) which contain a contiguous range of bytes into the modern file system comprised of a tree structure augmented with soft and hard links, sparse files, extended attributes and transparent support for many on-disk storage formats.

File systems perform many roles including storage of a user's data as well as meta data and configuration settings. When most users consider meta data that is stored by a file system they think of a file's size, modification time, access permissions etc. While such meta data has been in common use for a very long time, modern file systems allow much more meta data to be stored and retrieved¹. It has become common place for applications to store their configuration settings in the user's home directory under UNIX systems, extending the use of file systems to containing meta data about application instances themselves.

This paper has two distinct purposes: to survey current literature on the application of Formal Concept Analysis (FCA) to file systems and to present libferris² and how it offers new possibilities for application of FCA on file systems. It is assumed that the reader is familiar with the concepts of FCA. The common notation of the object set G , attribute set M and incidence relation $I \subseteq G \times M$ are used throughout.

The paper is organized as follows: A survey of preexisting efforts to apply FCA to file systems followed by an examination of the features of the virtual file system libferris that can be used for FCA. A brief conclusion then completes the paper.

¹ EA and ACL for Linux website, <http://acl.bestbits.at/>

² <http://witme.sourceforge.net/libferris.web/>

2 Formal Concept Analysis and File Systems

Ferré and Ridoux present an alternate representation of FCA as Logical Concept Analysis (LCA) in [7] where the lattice (G, M, I) has the attributes M replaced by an (possibly infinite) lattice of formulas and I attaches formulas to the objects in G .

Applying FCA to file systems can be seen from many perspectives in the literature. Much research has been done on applying FCA to text document collections [2,11]. FCA has also been applied to more structured data such as email [4]. Application to file systems as a whole is covered in [6].

There are many limitations of a hierarchical file system model which are addressed by using FCA. The most striking being that a tree structure forces logical containment of files in a directory and the encoding of meta data about each file into its path in the tree [4,6]. This can be eased by use of soft and hard links but in so doing the semantics of file access become more complicated (dangling links, cycles during link resolution). Encoding meta data into a file's path hinders the location of conceptually similar documents because a small change in one piece of meta data may require one to scan from the root of the tree to find a document. Consider the example where paths are created by first encoding the year the document was authored and then the general type of document such as audio, video or text. If one is browsing `"/2003/text/whitepaper/libferris/fcasurvey.tex"` and wishes to find other libferris whitepapers that were not necessarily authored in 2003 then they must try other branches from the root of the tree and scan down a similar path from each of those.

To alleviate the single access path issue many file systems offer the ability to find conceptually similar documents by showing the results of a query as a file system [8,9]. Such views have the drawback of being read only or allowing inconsistency and usually being somewhat separate from the standard navigation paths. In moving to the lattice structure of FCA both querying and navigation are presented via the same interface and a user can seamlessly switch between both styles of interaction [6].

In the upgrade from an hierarchical structure to a concept lattice, directories become concepts, symbolic links are no longer required and files form the object set G in the formal context. There is no requirement for symbolic links because FCA allows a file to exist in many concepts at the same time. Because a lattice structure allows a concept to have multiple parents it allows objects that are conceptually close to each other to be close in the lattice as well [1]. In the above example of looking for other libferris whitepapers one would only need to move up the lattice to loosen the restriction in the time dimension to see other related libferris papers. To gain access to informal documentation on libferris one could then navigate upward to remove the whitepaper attribute.

Ferré and Ridoux [6] generalize the current working directory `pwd(1)` into a history stack of working concepts. This is done to allow one to move to the correct parent concept easily. The familiar command `cd ..` becomes a pop operation on the history stack or a move to the root concept if the history is empty. The semantics of `cd ..` become more of a navigation backwards than

a navigation to the last direct superconcept as detailed in [6]. The change in semantics is because the relative or absolute move in the lattice is not broken into subparts and pushed individually but each refinement provides a single push onto the stack. If one is doing FCA using this working concept stack then one could break a navigation into its component attributes and push them as individual refinements. For example, given the working concept “/2003/text” and a command `cd whitepaper/libferris` the stack could have the two attributes pushed onto it in the order presented on the command line.

Due to concepts being multi parented there can be many paths from the root concept to any concept in the lattice. The “parent concept” stack should however still be maintained in the order given by the user so that only the last attribute in the path is removed by a `cd ..` command, ie. the particular absolute path chosen to find a concept is only relevant to future relative path operations. If one can easily strip off the last attribute of a path then one only needs to store the path used to reach the current concept to allow relative path operations. Presumably such a technique was not chosen for [6] due to the use of formulas for attributes in Logical Concept Analysis.

The `ls` command is made modal in [6] by separating out the query of the extent of a concept (`ls -r f`) from query for the refinements available from a working concept (`ls f`). This seems somewhat artificial as the traditional UNIX `ls` command makes no distinction between showing only places to navigate to against showing only the files in the current directory.

Using the working concept one can easily navigate the concept lattice using familiar commands `cd`, `ls` and `pwd` modifying the lattice using `mv`, `cp` and `rm` as has been done in the Conceptual Shell [6]. Although altering the current working directory with `cd` should be easy enough, explicit detail is not given in [6] about how one resolves copy, move and remove operations on objects in the lattice. The most challenging operation would be the `mv` command. Consider the case of moving an image from a subconcept of “true colour” to a subconcept of “monochrome”. Such an operation would require a lossy transformation to occur on the actual image data in order to maintain the semantic consistency of the objects in each concept in the lattice.

A commonly noted distinction is between intrinsic attributes of an object which may be mechanically determined from an object’s byte content, and extrinsic properties which require user interaction to determine. Extraction of intrinsic properties from documents covers many domain specific algorithms such as by using the Ripple-Down Rule (RDR) knowledge acquisition and maintenance methodology from knowledge based systems [11], email headers, regular expression matches and machine learning algorithms [4,15], or an arbitrary extraction function [6].

Extrinsic attributes for objects are discussed less than intrinsic. The Conceptual Email Manager (CEM) [4] uses extrinsic attributes to allow the user to override intrinsic attributes to always be **true** or **false** for a particular message and also to allow CEM to update attributes such as “mail read” and “new mail” [4].

The collection of intrinsic and extrinsic attributes is used to form the attribute set M for FCA. In CEM [4] the attributes create a partial order (M, \leq) such that the transitivity in the partially ordered attributes is also reflected in the relation I of the formal context (G, M, I) , ie. If for an object $g \in G$ and an attribute $m \in M$ if gIm then $\forall \mu \in \text{transitive-parent}(m), gI\mu$. This allows one to not only tag files with the most specific attributes but to find them in the formal context using less specific attributes relative to (M, \leq) . The partial order (M, \leq) in CEM is edited using a tree widget in which multi parented attributes appear under each of their parent attributes in the tree.

Although Ferré and Ridoux use formulas as their attributes they too apply an ordering to their attributes [7,6]. Their formulas are ordered by a possibly infinite lattice and they present methods to enable navigation of the concept lattice built from these formulas using contextualized logic.

Modern file systems support Extended Attributes (EAs) which allows arbitrary key-value data to be attached to files and directories. Additional APIs have been provided for both UNIX³ and Microsoft Windows⁴ operating systems for associating a key-value pair with a file. With EAs an application can store meta data about a file with the file itself on disk. One can abstractly consider the EAs for a file f as a subdirectory which can not contain directories but only files containing meta data about the file f . In this light, a directory can be seen as a many valued formal context. Assume that a directory with content G forms the objects g using its contents. A file $g \in G$ may have an EA $m \in M$ with value $w \in W$ where m and w are strings. Then w is functionally dependent on g and m .

Enrichments to FCA have been created to allow many valued attributes to be used and simple logics over attributes to generate summary attributes. Creating a binary relation that can be used as $I \subseteq G \times M$ can be done by creating conceptual scales [5,4,3,14] or using logical scaling [14,13]. Both conceptual and logical scaling can be seen as a method to take one or more columns in a many valued context and generate one or more new binary columns as the result.

CEM [4] creates conceptual scales automatically based on the partial order it maintains over the attributes in the formal context. A default scale S_μ is created $\forall \mu \in (M, \leq)$ such that S_μ is true iff an object g has any of the direct children attributes of μ in (M, \leq) . Using LCA for file systems [6] has a similar setup using the lattice of formulas it follows that one formula μ deduces all formula below it in the lattice.

3 libFerris and FCA

The libferris project was originally created to provide a modern semantic file system [8]. A semantic file system differs from a traditional file system in two major ways: by allowing the results of a query to be presented as a file system and to present interesting meta data from files as key-value attributes. Queries

³ EA and ACL for Linux website, <http://acl.bestbits.at/>

⁴ http://linux-ntfs.sf.net/ntfs/attributes/ea_information.html

are submitted to the file system embedded in the path and the results of the query form the content of the virtual directory. For example, to find all documents that have been modified in the last week one might read the directory `“/query/(mtime>begin last week)/”`. Interesting meta data is extracted from a file’s byte content using what are referred to as transducers in [8]. An example of a transducer would be a little bit of code that can extract the width of a specific image file format.

The core abstractions in libferris can be seen as the ability to present many trees of information overlaid into one namespace, the presentation of key-value attributes that files possess, a generic stream interface for file and attribute content, indexing services and the creation of arbitrary new files.

Overlaid trees are presented because one can drill into composite files such as XML, ISAM⁵ databases or tar files and view them as a file system. The overlaying of trees is synonymous with mounting a new file system over a mount point on a UNIX machine to join two trees into one globally navigable tree. Being able to overlay trees in this fashion allows libferris to provide a single file system model on top of a number of heterogeneous data sources⁶.

Presentation of key-value attributes is performed by either storing attributes in kernel level EAs or by creating synthetic attributes whose values can be dynamically generated and can perform actions when their values are changed. Both stored and generated attributes in libferris are referred to simply as EAs. Examples of EAs that can be generated include the width and height of an image, the bit rate of an mp3 file or the MD5⁷ hash of a file. For an example of a synthetic attribute that is writable consider an image file which has the key-value EA `width=800` attached to it. When one writes a value of `640` to the EA `width` for this image then the file’s image data is scaled to be only 640 pixels wide. Having performed the scaling of image data the next time the `width` EA is read for this image it will generate the value `640` because the image data is 640 pixels wide. In this way the differences between generated and stored attributes are abstracted from applications.

Another way libferris extends the EA interface is by offering schema data for attributes. Such meta data allows for default sorting orders to be set for a datatype, filtering to use the correct comparison operator (integer vs. string comparison), and GUIs to present data in a format which the user will find intuitive. Having the correct comparison operator and sorting order is a prerequisite to generating logical scales for an EA.

Although attaching and interacting with typed arbitrary key-value data on files is very convenient in libferris it leaves applications open to interpret the data how they choose. For this reason specific EAs have been defined for semantic categorization of files on a system wide basis. These EAs allow one to associate

⁵ Indexed Sequential Access Method. eg. B-Tree data stores such as Berkeley db.

⁶ Some of the data sources that libferris currently handles include; http, ftp, db4, dvd, edb, eet, ssh, tar, gdbm, sysV shared memory, LDAP, mbox, sockets, mysql, tdb and XML.

⁷ MD5 hash function RFC, <http://www.ietf.org/rfc/rfc1321.txt>

files with “emblems” to explicitly capture the intrinsic and extrinsic attributes of a file. The set of all emblems ξ is maintained in a partial order (ξ, \leq) . The relation for $\mu, \varphi \in \xi$ of $\mu \leq \varphi$ means that μ logically **is-a** φ . Consider the example of marking an image file: one may attach the emblem “sam” to the image file. This emblem may have a parent “my friends” to indicate that sam is one of my friends. It follows that the image file is also of one of my friends. A file can have many emblems attached to it. For best querying results the most specific emblems from (ξ, \leq) that are applicable to a file should be attached.

In a way the partial order of emblems (ξ, \leq) maintains a richer structure than the simple directory inclusion used in standard hierarchical file systems. If we are to define direct directory inclusion as a relation consider that we have a set of documents G arranged into a standard file system tree using the relation $\lambda \subset G \times G$. The relation λ is considered as anti-transitive, i.e. $x\lambda y, y\lambda z \Rightarrow \neg(x\lambda z)$. The relation λ is also not reflexive and is antisymmetric. One can use λ to represent the normal parent relationship from file systems so: $a\lambda b \Leftrightarrow a$ is a direct parent of b . In normal file systems each object would have only one parent.

The emblems (ξ, \leq) can be mounted as a file system in libferris and allow retrieval of all objects that have an emblem when the leaf emblems in (ξ, \leq) are listed. Thus executing `ls -l emblems://whitepaper/ferris` would show a list of all files that have been tagged with **ferris** or any of its transitive parent emblems in (ξ, \leq) .

In order to quickly find the set of files that have a given attribute value indexing is available on both the full text of files [17] and on their EAs. The EA index is maintained in a sorted order to allow the list of files for which a comparative constraint on their EA holds. For example `width<800` can be resolved completely using only the index.

Given the indexes, files, emblems and attributes from libferris there are many ways to create a formal context: logical scaling, file system inclusion and the presence of emblems.

The most mechanical of these is using emblems because a file either has one attached or it doesn't. Logical scaling can be used by supplying a simple boolean logic to use on the attribute index, for example, `width < 900`. There are several predicate languages used at current: an extended Lightweight Directory Access Protocol (LDAP) search syntax⁸ and XPath expressions⁹.

File System inclusion is a form of logical scaling which when given a file system Z will create a new attribute ζ in the context. For each $g \in G$ The value of $gI\zeta$ will be true iff $\exists \mu \in Z$ such that $\mu = g$. Using file system inclusion and the full text index one can define an attribute in the formal context which is true iff objects $g \in G$ satisfy a boolean full text query. For example, if we wish to have a new attribute ζ indicating if an object satisfies the boolean query **alice wonderland** we first mount the full text query and obtain the file system Z showing the files containing these terms and then bind that file system to the formal context using file system inclusion

⁸ <http://www.faqs.org/rfcs/rfc2254.html>

⁹ XML Path Language (XPath) Version 1.0, <http://www.w3.org/TR/xpath>

$$\mathbb{K} = (G, M \cup \{\zeta\}, I \cup \{(g, \zeta) | g \in G, \exists \mu \in Z, g = \mu\}) \quad (1)$$

One of the main utilities of this scheme is to allow existing virtual file systems that libferris can generate based on queries to be leveraged in FCA.

In a way the use of emblems to categorize ones files combined with the use of a properly synchronized index can be seen as a mechanism for cached queries. In this light it is possible to allow logical scaling to be performed before it is used in FCA. For example, by using a collection of emblems suitably parented one can cache numeric intrinsic results. This is done using a chain of emblems $a \leftarrow b \leftarrow c \leftarrow d$ where the description of a is “< 10” and b is “< 5” and so on. A script can then be run to operate over a file system tree and assign the most specific emblem to each file for latter use in FCA. This does imply that the level of quantization captured in the emblem chain is acceptably small when the objects are tagged with emblems to allow future use in FCA.

There are three main possibilities as to who attaches emblems to files: user explicit assertion or retraction, use of rigid rule sets (such as *width* ≥ 1024 implies attachment of the **medium-resolution-image** emblem) and use of Supervised Machine Learning (SML) algorithms. Automatic attachment has been explored using SML algorithms such as Support Vector Models [10] or Bayesian networks¹⁰. The main issue with using such SML algorithms is that they are not entirely accurate. See [12] for further examination of emblem attachment and in particular automated emblem attachment.

4 Conclusion

A survey of the limited work that exists applying Formal Concept Analysis to file systems and semi structured data has been presented. A modern semantic file system, libferris, which can be used to generate formal contexts suitable for FCA has been presented. Discussion of libferris covered the use of EAs to create many valued formal contexts and the use of emblems as a direct source of binary attributes. Some concerns such as the requirement of indexing were also touched on.

By using a modern semantic file system as the data source for FCA one extends the horizons of what they can analyze to include not only structured data from relational databases [16], semi structured email data [4] or full text documents [2] but data from many sources including directly from the Internet.

References

1. Peter Becker and Peter Eklund. Prospects for formal concept analysis in document retrieval. In *Australian Document Computing Symposium (ADCS01)*, pages 5–9. University of Sydney, Basser Department of Computer Science, 2001.
2. C. Carpineto and G. Romano. Order-theoretical ranking. *Journal of the American Society for Information Science (JASIS)*, 51(7):587–601, 2000.

¹⁰ bogofilter homepage, <http://bogofilter.sourceforge.net/>

3. Richard Cole. *Document Retrieval using Formal Concept Analysis*. PhD thesis, School of Information Technology, Griffith University, 2001.
4. Richard Cole, Peter Eklund, and Gerd Stumme. Document retrieval for email search and discovery using formal concept analysis. *Applied Artificial Intelligence*, 17(3), 2003.
5. Richard Cole, Peter Eklund, and Don Walker. Constructing conceptual scales in formal concept analysis. In *Proceedings of the 2nd Pacific Asian Conference on Knowledge Discovery and Data Mining*, number 1394 in LNAI, pages 378–379. Springer Verlag, 1998.
6. Sebastien Ferré and Olivier Ridoux. A file system based on concept analysis. In *Computational Logic*, pages 1033–1047, 2000.
7. Sebastien Ferré and Olivier Ridoux. A logical generalization of formal concept analysis. In Guy Mineau and Bernhard Ganter, editors, *International Conference on Conceptual Structures*, August 2000.
8. David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, and James W. Jr O’Toole. Semantic file systems. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, ACM SIGOPS, pages 16–25, 1991.
9. Burra Gopal and Udi Manber. Integrating content-based access mechanisms with hierarchical file systems. In *Proceedings of third symposium on Operating Systems Design and Implementation*, USENIX Association, pages 265–278, 1999.
10. T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.
11. M. Kim and P Compton. Formal concept analysis for domain-specific document retrieval systems. In *The 13th Australian Joint Conference on Artificial Intelligence (AI’01)*, pages 179–184. Springer-Verlag, 2001.
12. Ben Martin. File system wide file classification with agents. In *Australian Document Computing Symposium (ADCS03)*. University of Queensland, 2003.
13. S. Prediger. Symbolic objects in formal concept analysis. In *Proceedings of the Second International Symposium on Knowledge, Retrieval, Use and Storage for Efficiency*, 1997.
14. Susanne Prediger. Logical scaling in formal concept analysis. In *International Conference on Conceptual Structures*, pages 332–341, 1997.
15. Mark Rosen. E-mail classification in the haystack framework, 2003. MIT, Masters thesis, describes the design and implementation of a text classification framework for the Haystack project.
16. Frank Vogt and Rudolf Wille. Toscana — a graphical tool for analyzing and exploring data. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing*, pages 226–233, Heidelberg, 1995. Springer-Verlag.
17. Ian H. Witten, Alistar Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 340 Pine Street, San Francisco, CA 94104-3205, USA, 1999.

Numerical Analysis in Conceptual Systems with TOSCANAJ

Peter Becker

KVO Laboratories
316/38 Warner Street
Fortitude Valley
QLD 4006, Australia
`peter@peterbecker.de`

Abstract. TOSCANA-systems have been used in many contexts to visualize information found in relational databases. As opposed to many other approaches this is done based on conceptual structures rather than numerical analysis. While conceptual structures often tend to ease understanding of the data by avoiding too much information reduction, sometimes a particular reduced information gained by using numerical approaches is preferable. This paper discusses how conceptual structures and numerical analysis can be combined into conceptual information systems using TOSCANAJ.

1 Introduction

The development of the theory of conceptual information systems always went along with the usage and development of tools, most noticeably TOSCANA. As the first program of its kind, TOSCANA has a long history of projects where it was applied, but it also encountered limitations due to aging effects of its code base. About 2 years ago the KVO group¹ started the TOSCANAJ project² with the aim to address these issues with a new architecture. This work is based on the experiences from TOSCANA projects, mostly coming from the Technical University in Darmstadt and it aims at giving both a professional tool for applied conceptual knowledge processing as well as a platform to extend the theory of conceptual information systems.

One area where TOSCANAJ offers a range of new features extending the existing notion of conceptual information systems is the integration of numerical methods. In combination with a relational database management system a range of new combinations of conceptual and numerical analysis and visualization can be achieved, such as:

- labels in line diagrams can display numerical aggregates of object sets;
- classical textual database reporting views can be displayed within the program, thus easing access to larger amounts of numerical analysis data;

¹ <http://www.kvocentral.org>

² <http://toscanaj.sourceforge.net>

- charts can be accessed for any object set in a displayed lattice;
- ratios of object sets can be displayed – in comparison to neighbouring sets or the full object set within a diagram

Some of these features have been described in earlier work, most noticeably by Stumme and Wolff (see e.g. [SW97], [SW98]). This paper tries to give an overview of the features in TOSCANAJ relating to numerical analysis and to contextualise these in a larger framework based on the ideas of Stumme and Wolff. The implementations in TOSCANAJ are used as examples and proof of concept – unless otherwise stated all the described functionality has been implemented and used in conceptual information systems using TOSCANAJ.

The paper is structured as follows: Section 2 gives some background in form of an overview of the literature. Section 3 discusses the use of aggregates as label contents in a line diagram. Section 4 shows how reporting tools similar to standard database analysis tools can be used within a conceptual information system. Section 5 discusses how to attach comparisons of extent sizes as labels onto the lines. The paper concludes with an outlook for further developments in Section 6.

2 Background

We assume the reader is familiar with Formal Concept Analysis and the terminology and notations introduced by Ganter and Wille in [GW99]. We also assume that the reader knows about the notion of conceptual information systems as defined for example in [HSWW00]. An overview of TOSCANAJ and its features can be found in [BH04].

While the notion of a conceptual information system has been well elaborated in the literature ([HSWW00], [HS01], [Stu00]), the combination with numerical methods is less well researched – with the exception of the combination of FCA and data mining (e.g. [STB⁺02]).

Stumme and Wolff have presented two distinct ideas in some of their papers ([SW97], [SW98]): the use of aggregating functions to extend conceptual information systems to integrate so-called *relational structures* as extension to the idea of many-valued contexts and the use of Pearson's χ^2 -measure to determine the dependency of two scales.

Data mining and FCA are supported by a range of tools like the TITANIC algorithm ([STB⁺02]) and different programs. A tool called CHIANTI was written by Stumme and Hereth ([HSWW00]) to investigate different measures for the distance of scales³. Other aspects like the combination of aggregating functions with conceptual systems have been described in research papers, but to the knowledge of the author never been implemented.

This paper will discuss some of these combinations and how they are implemented in the TOSCANAJ program. Beside giving a proof of concept with the implementation, some additional notions like the integration of reporting tools

³ note that we use “distance” in a wider sense than metric.

will be discussed to extend the notion of a conceptual information system into a broader data analysis technique.

3 Displaying Aggregates within the Line Diagrams

TOSCANA 2 and 3 have two basic options for aggregation – instead of just allowing the user to see the lists of the items in the extents or object contingents, they also allow one to show the object counts either as absolute numbers or as distribution, that is, relative to the size of the current object set. For many conceptual information systems created this was sufficient, but for other systems it turned out to be a limitation since other aggregates were considered useful in their context but not easily available.

While using the same default options for the labels, TOSCANAJ tries to give the Conceptual System Engineer more flexibility by allowing the definition of additional label contents in the CSX file, which contains the schema for the conceptual system in XML format. Thus the user of the system will be offered more options for the labels if the corresponding schema is loaded. These options can be either lists or aggregates. Since we are interested in the numerical aspects only the latter will be discussed here.

If labels should display aggregates, the CSX file has to contain a definition of an `<aggregateQuery>`, which contains at least one SQL fragment defining the aggregate to use. All aggregates are defined on the SQL level, which gives a great level of flexibility and good performance since the aggregation happens in the database engine. Most database engines also allow writing extensions, in this way new aggregates can be introduced. By not using a separate aggregation system, performance and a clean abstraction is achieved.

Not only can the Conceptual System Engineer use any aggregate defined in the underlying database management system, TOSCANAJ also offers two orthogonal extensions:

- aggregates can be formatted (e.g. as currencies) and multiple aggregates can be combined into a single label; and
- the results can be viewed as relative numbers compared to the same aggregate on all objects in the current diagram.

These extra options can be illustrated using two examples. The first one gives the price range of the objects in question by querying the minimum and maximum of the `PRICE` column and displaying it in a formatted way as shown in Fig. 1 on the left. A result of this query can be seen to the right. The definition is given only structurally to avoid the overhead of the XML syntax. In the CSX file the indentation will be given as nested XML elements with attributes. The name is used for menu entries.

Displaying the aggregate relative to the same aggregate on the current object set is a generalization of the distribution labels from TOSCANA 2/3. There the number of objects in a certain set (extent/contingent) is displayed relative to the full number of objects. A similar approach can be taken with other aggregates,

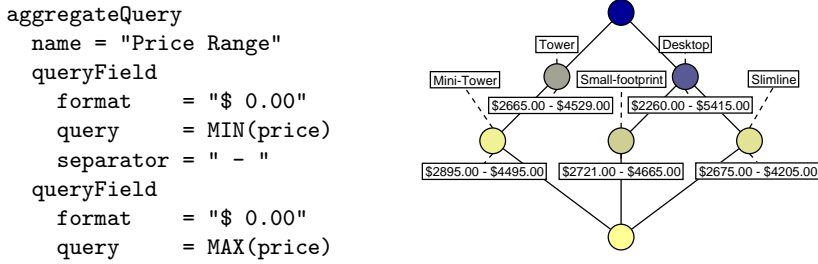


Fig. 1. Querying a range from a database column

although of course the interpretation changes. For example the query shown in Fig. 2 queries the average price once as normal query, and then again as a relative value, compared to the average on the current object set. The value of 98.93% on the right in the resulting diagram means that the average price of the objects in the contingent is 1.07% below the average of the objects displayed in the diagram (either the full object set or the set reduced by filtering).

These features of using SQL aggregates, being able to combine them and the relative views allow a conceptual information system to be customized to display very specific information useful for it. The Conceptual System Engineer can mix numerical analysis into the conceptual information system in a suitable fashion to enhance the ease of use and usefulness of the system created.

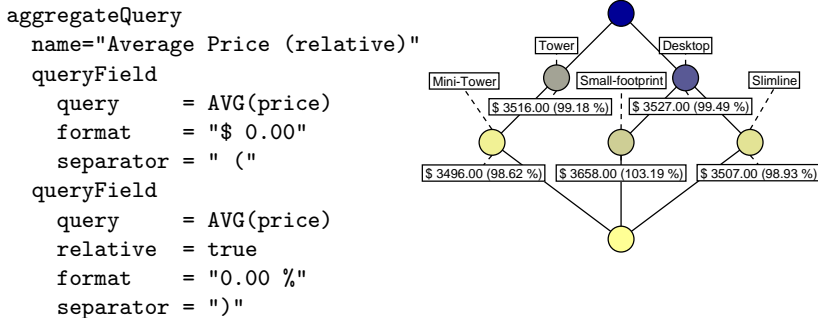


Fig. 2. Querying an average value as absolute and relative numbers

4 Using Conceptual Structures to Access Numerical Reports

If more than just simple aggregates are needed to be shown to the user, displaying this information directly in the line diagram is not feasible anymore. In this case additional windows are needed that can be shown on request. TOSCANAJ follows

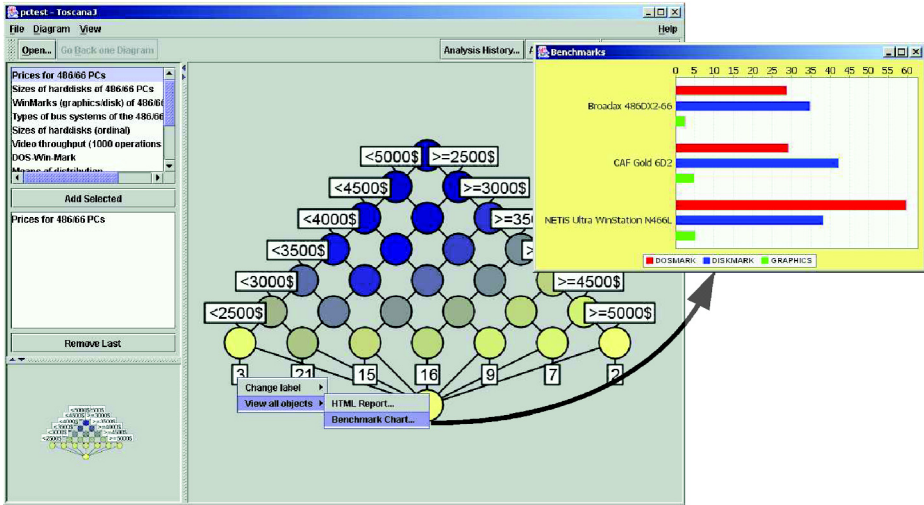


Fig. 3. Calling a bar chart to display numerical information about an extent

the tradition of the older TOSCANAJ systems by attaching this operation to the object labels – since the information is about the object sets, this is a well accepted approach.

There are two distinctive popup windows the user can open: the first are views on single items, which provide information about a particular object in a set. To access this object-specific information, a particular object has to be selected from a list. As opposed to this, reports showing information about a full object set can be selected on any type of label.

Obviously the reports allow integration with aggregates and other numerical facilities. TOSCANAJ 2 and 3 used MS Access forms and HTML templates to do the reporting, while TOSCANAJ offers a plugin interface with a number of implementations. One of them uses a similar approach to TOSCANAJ 3 and uses HTML templates to create reports. As opposed to TOSCANAJ 3, TOSCANAJ can show these as windows within the main application – TOSCANAJ 3 had to start an external browser. The internal approach allows better use of the screen real estate.

TOSCANAJ exceeds the functionality of the older programs in its ability to access charts for the object sets in a lattice. Due to the integration with the JFreeChart rendering engine⁴, a range of different charts can be displayed within the framework of the scales. This includes line charts, bar charts, pie charts, area charts, scatter plots and many more. The charts are highly configurable and additional information like moving averages can be added into the charts.

Fig. 3 is a screenshot of TOSCANAJ showing a bar chart of benchmark data for a particular extent. Via the context menu of the leftmost object label the

⁴ <http://jfree.org/jfreechart>

chart option configured in the CSX file has been selected, which opens a chart created by JFreeChart. This chart shows the results for three different types of benchmark in a quick overview. For this type of information a chart is far more efficient than a conceptual scale and the integration allows integrating typical views from existing data analysis approaches into a conceptual information system.

This combination of conceptual and numerical data analysis allows the user to combine the two approaches to synergetic effects. While the conceptual structure gives the guidance to identify the objects of interest, the numerical summary information in the reports – either in textual or in chart form – allows quickly identifying certain characteristics of the selected object set. These may then in turn lead to more refinement within the conceptual information system.

5 Looking at Object Frequencies

So far we have looked at numerical information we display based on single object sets – with the exceptions of the relative aggregates where the values found get normalized. Sometimes it is interesting to look at the aggregates for different sets and how they compare to each other. In our work on TOSCANAJ we have so far investigated only the comparison of the object counts with each other, since this is the simplest case. Some of this work should be extended to other aggregates.

One of our ideas of visualizing the ratios of object counts is to look at the extent ratios of neighbouring concepts. This measure gives a notion close to the notion of “support” in data mining. The extent ratio between two concepts $C_1 \succ C_2$ tells how many objects which have all attributes in the intent of C_1 will also have the additional attributes in the intent of C_2 .

These ratios can be visualized in TOSCANAJ as shown in Fig. 4. This example shows how easy it is to read information like “about 80% of the PCs are available via direct sales” or “about two thirds of the PCs available in shops are also available via direct sales”. Since this type of information is about a ratio of extents the information is easier to access on the edges.

6 Outlook and Conclusion

We have shown a number of different ways to combine the structuring power of conceptual information systems with the information reduction of numerical approaches. Since the TOSCANAJ features presented in the paper are mostly set up for specific conceptual information systems, the numerical aspects can be applied in a well-targeted fashion, based not only on the informational structure found in the data source, but also on the abilities of the end-user.

TOSCANAJ is an ongoing project and it is likely that more features in the direction described here will be added in the near future. Another planned step to allow more flexibility and power in configuring tailored information systems

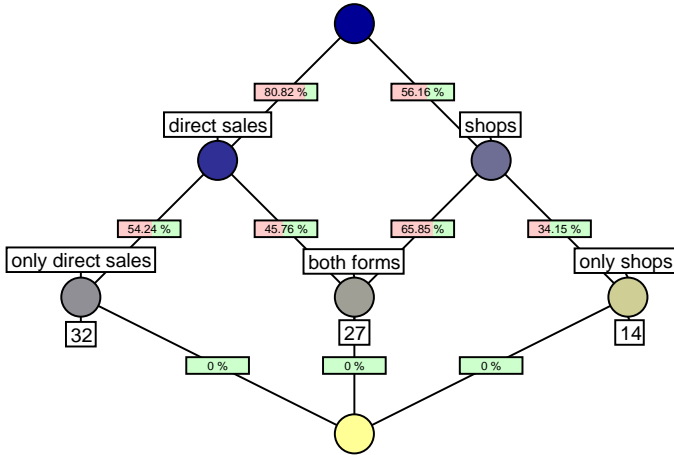


Fig. 4. Labels on the edges denote extent ratios

is the unification of the different views used for displaying different types of information.

Except for the labels on the lines all views visualize certain aspects of the concepts. This includes not only the object and attribute labels and the separate pop-up windows as used by the reporting tools, it also includes the nodes themselves since in TOSCANAJ they do not just denote the concepts' positions as in a classical line diagram, they also convey information about the concepts. In the standard setting this is a color gradient hinting at the extent size.

All these views – labels, pop-ups and nodes – can be seen in some sense as interchangeable. Most interesting seems to be the idea of replacing the standard nodes with more informative structures like charts or UML-like boxes showing attribute and object contingents. Furthermore TOSCANAJ could allow an arbitrary number of labels per concept and the notion of pop-up windows could be unified with the labels by maintaining a connector as long as the pop-up is visible. This would highly increase readability when multiple pop-up windows are open.

Such a refactoring of TOSCANAJ would give great flexibility to advanced Conceptual System Engineers. While we still propose to keep the default configurations rather simple for ease of use, the option to tailor the type and amount of information displayed in a particular TOSCANAJ system seems to be a very worthwhile goal to pursue – both for research purposes as well as discovering new ways of applying conceptual information systems in real world projects.

Acknowledgement

Part of this work was paid by DSTC Pty Ltd, Brisbane, Australia.

References

- BH04. Peter Becker and Joachim Hereth Correia. The ToscanaJ suite for implementing Conceptual Information Systems. In *Formal Concept Analysis – State of the Art*. Springer, Berlin – Heidelberg – New York, 2004. To appear.
- GW99. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin – Heidelberg – New York, 1999.
- HS01. Jo Hereth and Gerd Stumme. Reverse pivoting in Conceptual Information Systems. *Lecture Notes in Computer Science*, 2120:202–215, 2001.
- HSWW00. Joachim Hereth, Gerd Stumme, Rudolf Wille, and Uta Wille. Conceptual knowledge discovery and data analysis. In *International Conference on Conceptual Structures*, pages 421–437, 2000.
- MS99. Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- STB⁺02. Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with Titanic. *Data Knowledge Engineering*, 42(2):189–222, 2002.
- Stu00. Gerd Stumme. Conceptual on-line analytical processing. In S. Ghandeharizadeh K. Tanaka and Y. Kambayashi, editors, *Information Organization and Databases*, chapter 14, pages 191–203. Kluwer, Boston–Dordrecht–London, 2000.
- SW97. Gerd Stumme and Karl Erich Wolff. Computing in conceptual data systems with relational structures. In *Proceedings of KRUSE '97*, pages 206–219, 1997.
- SW98. Gerd Stumme and Karl Erich Wolff. Numerical aspects in the data model of conceptual information systems. In *ER Workshops*, pages 117–128, 1998.

Tool Support for FCA

Thomas Tilley

School of Information Technology and Electrical Engineering
University of Queensland, Brisbane, Australia
tilley@itee.uq.edu.au

Abstract. Over the last two decades a number of tools have been developed to support the application of Formal Concept Analysis (FCA) to a wide variety of domains. This paper presents an overview of tool support for FCA.

1 Introduction

Over the last two decades a number of tools have been developed to support the application of Formal Concept Analysis (FCA) to a wide variety of domains. These tools range from the early DOS-based implementations of Duquenne's GLAD tool to Java-based tools currently under active development like ConExp and ToscanaJ. Both commercial and open-source software appears in the list which also includes general-purpose and application specific tools.

The next section of the paper introduces the general purpose tools. Application specific tools are then discussed in Section 3 before Section 4 concludes the paper.

2 FCA Tools

Duquenne's tool for General Lattice Analysis and Design (GLAD) is possibly the earliest software tool that facilitates the analysis of formal concept lattices [1]. GLAD is a DOS-based program written in FORTRAN that has been under development since 1983. The tool facilitates the editing, drawing, modifying, decomposing and approximation of finite lattices in general and is not restricted to the analysis of concept lattices. The lattices to be analysed can be derived from abstract mathematics or applied statistics using techniques like Analysis of Variance. Single-valued data can also be analysed by exploiting the classic correspondence between lattices and binary relations identified by Birkhoff.

GLAD contains a large number of features, many of which are undocumented and it also supports "scenarios" which represent a form of macro. These scenarios can be used regenerate and manipulate a lattice by recalling the list of commands used to construct it. Diagrams can also be output directly from GLAD in the Hewlett Packard Graphics Language (HPGL)¹ — a vector based language designed for plotters.

ConImp (**C**ontexts and **I**mplications) is another DOS-based tool implemented by Peter Burmeister [2] who started development in 1986 on an Apple II computer. While ConImp is purely text based and provides no graphical output for lattices it also supports

¹ See <http://www.piclist.com/techref/language/hpgl.htm>

a wide range of features for manipulating contexts and provides concept listings which can be used for drawing line diagrams by hand.

The *Duquenne-Guigues-base* represents a canonical base of valid implications for a given context and this is computed and used extensively within ConImp. Interactive attribute exploration is supported which can be used to derive both the Duquenne-Guigues-base and a typical set of objects. In addition, a three-valued logic that allows for *true*, *false* and *unknown* values can also be used.

While ConImp supports single-valued contexts another tool called *MBA* (possibly from the German for “Many-valued FCA”: “*Mehrwertige BegriffsAnalyse*”) can be used to scale and pre-process many-valued contexts². In addition, contexts can be exported from ConImp in the so called “Burmeister Format” (‘.CXT’) and rendered using another DOS-based tool called *Diagram* [3]. The use of separate tools for the tasks of data-preparation, context creation, and line diagram rendering (*Diagram*) is also reflected in the classic FCA tools Anaconda and TOSCANA.

Anaconda and TOSCANA (**TO**ols of **C**oncept **A**NALysis) are tools used for building conceptual knowledge systems on top of data stored in relational databases. A conceptual system engineer uses knowledge from a domain expert to create queries in the form of conceptual scales using a *conceptual system editor*. These scales essentially capture the expert’s knowledge and the information is stored in a *conceptual system file*. A user can then exploit the conceptual scales to retrieve or analyse data from the database using a *conceptual schema browser* [4]. In traditional TOSCANA systems Anaconda is the conceptual system editor, TOSCANA is the conceptual system browser, and the data is stored in a Microsoft Access database.

Anaconda is a tool for the creation and editing of contexts, line-diagrams and scales. The context, scales and line-diagrams are saved in a conceptual schema file which is then used by TOSCANA to analyse the data in the database. While TOSCANA users cannot create new scales, the scales can be composed to produce nested line diagrams. There are three versions of TOSCANA based on Vogt’s C++ FCA libraries [5,6] and more recently a Java-based version — ToscanaJ.

ToscanaJ [4] is a platform-independent implementation of TOSCANA that supports nested line diagrams, zooming and ideal/filter highlighting. Originally part of the Tockit project³ — an open source effort to produce a framework for conceptual knowledge processing in Java — ToscanaJ is now a separate project⁴.

In the context of the workflow described earlier, ToscanaJ represents the conceptual schema browser and the conceptual system editor role is filled by two tools — *Siena* and *Elba*. The two tools can be seen as Anaconda replacements that are both used for preparing contexts and scales, however, each represents a different workflow. Elba is used for building ToscanaJ systems on top of relational databases while Siena allows contexts to be defined using a simple point and click interface.

ToscanaJ can be used to analyse data in relational databases via ODBC (Open Database Connectivity)/JDBC (Java Database Connectivity) or, alternatively, an embedded relational database within ToscanaJ can be used. Line diagrams can also be

² See <http://www.mathematik.tu-darmstadt.de/ags/ag1/Software/>

³ See <http://tockit.sourceforge.net/>

⁴ See <http://toscanaj.sourceforge.net/>

exported in a variety of raster and vector-based formats including Portable Network Graphics (PNG), Joint Photographic Expert Group (JPEG), Encapsulated PostScript (EPS), Portable Document Format (PDF), and Scalable Vector Graphics (SVG).

An XML-based conceptual schema file (.CSX) is used to store the context and scales produced by Siena and Elba. In addition, an extensible viewer interface allows custom views to be defined as well as allowing external data viewers to be specified. This feature is exploited by the formal specification browser SpecTrE described in Section 3.2.

The line diagrams in Siena and Elba use an n -dimensional layout algorithm in which each attribute in the purified context is assigned to a vector [7]. The layout is then projected onto the Cartesian plane using standard parallel projection and the approach is based on the algorithm used in Cernato.

Cernato is a commercial FCA tool developed by Navicon⁵ that combines some of the features of Anaconda and TOSCANA into a single tool. Users are presented with a familiar spreadsheet-like interface for creating contexts and data can be imported and exported in Comma Separated Value (CSV) format which facilitates the analysis of data from genuine spreadsheet applications.

Line diagrams are constructed incrementally in Cernato and the layout is animated by default. Zooming and the construction of scales, which are known as “views” in Cernato, are also supported, however, nested line-diagrams are not. In addition to the CSV import/export facility a custom XML format can also be used. Furthermore, line diagrams can be exported in a number of raster-based image formats, contexts can be saved as HTML-tables and Cernato is also able to export complete TOSCANA systems.

ConExp (**C**oncept **E**xplorer)⁶ is another Java-based, open-source FCA project. Like Cernato, ConExp combines context creation and visualisation into a single tool. While ConExp does not support database connectivity, contexts can be imported and exported in ConImp’s ‘.CXT’ format. A number of lattice layout algorithms can be selected including chain decomposition and spring-force algorithms. The line diagrams also support various forms of highlighting including ideal, filter, neighbour and single concept highlighting and can be exported in JPEG or GIF format.

ConExp currently implements the largest set of operations from Ganter and Wille’s FCA book [8] including calculation of association rules and the Duquenne-Guigues-base of implications. Interactive attribute exploration is also supported and the context can display the arrow relations $g \nearrow m$ and $g \searrow m$.

GaLicia⁷, the Galois Lattice Interactive Constructor is another Java-based FCA tool that provides both context creation and visualisation facilities [9]. GaLicia’s heritage lies in a series of incremental data mining algorithms originally entitled the **G**ALOIS **L**ATTICE-BASED **I**NCREMENTAL **C**LOSED **I**TEMSET **A**PPROACH and also a *trie* data-structure based version called GALICIA-T. These incremental algorithms were used for mining association rules in transaction databases [10,11] and form the basis for the incremental construction of lattices in GaLicia.

Both single and many-valued contexts can be analysed in GaLicia. In addition, binary relationships between objects can also be described via a context and stored

⁵ See <http://www.navicon.de>

⁶ See <http://sourceforge.net/projects/conexp>

⁷ See <http://www.iro.umontreal.ca/~valtchev/galicia/>

using GaLicia's Relational Context Family [12]. A number of different lattice and Galois sub-hierarchy construction algorithms are also supported.

GaLicia provides two lattice layout mechanisms including a “magnetic” spring-force algorithm. The lattices can also be viewed using a novel, rotating 3-Dimensional view. GaLicia can be run as a stand-alone application or it can be used via the World Wide Web as a Java applet running in a Web browser.

3 Other Tools

Having introduced a number of generic FCA tools in the preceding sections, this section provides a brief overview of some application specific FCA tools. These tools can be broadly classified into two main groups: the *modular* and the *monolithic*. Tools that rely on other programs for part or all of their functionality will be classified as “modular”. For example, a number of the application specific tools make use of pre-existing graph drawing applications for lattice layout. In contrast the term “monolithic” will be used to describe those tools which do not rely on other applications to function. This does not, however, exclude the use of pre-existing libraries within the tools code. Additionally, the term should not infer that a tool is poorly engineered or necessarily massive, but rather that the tool has been constructed from scratch.

3.1 Monolithic Approaches

Düwel's *BASE* [13] tool supports the identification of class candidates from use-cases. The name is taken from the German “*ein Begriffsbasiertes Analyseverfahren für die Software-Entwicklung*” which translates into English as “concept-based analysis during software development”.

Taran and Tkachev's [14] tool SIZID is designed to support the analysis of sociological and psychological data. SIZID can handle multi-valued contexts and the calculation of implications.

Cole and Eklund have implemented a number of FCA based document management and information retrieval tools. *Warp-9 FCA* [15] is a tool for managing a collection of medical discharge documents that is implemented using the scripting and extension language Tcl/Tk⁸. A medical ontology to index documents and the visualisation supports folding line diagrams. The ideas in *Warp-9 FCA* are further refined and applied to the analysis of email in the tool CEM — the Conceptual Email Manager [16]. More recently a commercial descendant of CEM known as *Mail-Sleuth* has also been released⁹.

In Lindig and Snelting's [17] paper on the structure of legacy code a footnote mentions an inference based software environment called NORA which was used to produce the analyses described in the paper. NORA stands for “NO Real Acronym”. While no details of the NORA environment are presented in the paper, both Snelting and Lindig have produced other tools to support the analysis of software using FCA. Snelting and Streckenbach's *KABA* is a Java-based tool that implements the analysis earlier described

⁸ See <http://www.tcl.tk>

⁹ See <http://www.mail-sleuth.com>

by Snelting and Tip [18]. The name KABA is taken from the German “KlassenAnalyse mit BegriffsAnalyse” which translates as “class analysis via concept analysis” in English. Apparently “KABA” is also the name of a popular chocolate drink in Germany.

KABA combines concept lattices with dataflow analysis, and type inference. In particular the prototype tool supports the visualisation of horizontal-decompositions in Java classes and a 15 KLOC (“thousand Lines Of Code”) example is reported.

While another prototype tool that implements Lindig’s component retrieval ideas could be considered monolithic [19], there have been a number of modular tools developed using Lindig’s *concepts*¹⁰ framework.

3.2 Modular Approaches

Concepts is an updated version of Lindig’s *TkConcept* tool¹¹ implemented in Tcl/Tk. *TkConcept* is included here as an example of a modular tool because it makes use of a graph layout application called *Graphplace*¹² to draw lattice diagrams. *TkConcept* was intended as a framework for concept analysis applications that provides basic abstractions so that software designers can focus on the implementation of domain specific parts of an application.

Van Deursen and Kuipers [20] used Lindig’s *concepts* tool in conjunction with *Graphplace* in the analysis of a 100 KLOC COBOL program. A relational database was used to derive information about the application using a COBOL lexical analysis tool. The data was then extracted and formatted for analysis with *concepts*.

The *ConceptRefinery* tool described by Kuipers and Moonen [21] also uses *concepts* in conjunction with a COBOL parser and a relational database. Concept refinery is implemented using Tcl/Tk and a version of the *dot* directed graph drawing tool was used for visualisation. *Dot* is part of the *GraphViz* graph visualisation package¹³.

GraphViz and *concepts* are also used to render lattice diagrams in Eisenbarth et al.’s *Bauhaus* tool [22]. *Bauhaus* makes use of a number of components including the *gcc* compiler and *gprof* profiler which are glued together using Perl. In addition to their earlier work identifying features in web-browser code, Eisenbarth et al. have also used their tool to analyse a 1,200 KLOC production system.

The *Cable* tool implemented by Ammons et al. makes use of FCA to aid in the debugging of temporal specifications [23]. This visualisations presented to *Cable* users are implemented using the *Dotty* and *Grappa* graph visualisation tools which are also part of *GraphViz*.

Janssen’s *JaLaBA* tool¹⁴ is a novel on-line Java Lattice Building Application that uses Freese’s *LatDraw*¹⁵ program for lattice layout. *LatDraw* makes use of a 3-dimensional spring and force layout algorithm which produces line diagrams similar to *GaLicia* and *ConExp*.

¹⁰ See <http://www.eecs.harvard.edu/~lindig/src/concepts.html>

¹¹ See <http://sensei.ieec.uned.es/manuales/tkconcept/welcome.html>

¹² Available from

<ftp://ftp.dcs.warwick.ac.uk/people/Martyn.Amos/packages/graphplace/graphplace.tar.gz>

¹³ See <http://www.research.att.com/sw/tools/graphviz/>

¹⁴ See <http://juffer.xs4all.nl/cgi-bin/jalaba/JaLaBA.pl>

¹⁵ See <http://www.math.hawaii.edu/~ralph/LatDraw/>

The round-trip engineering work of Bojic and Velasevic [24] also makes use of a modular tool implemented using ConImp. By adapting the output from the Microsoft Visual C++ profiler, ConImp was able to analyse their data which was then used to update a UML model using the Rational Rose design tool¹⁶.

Richards and Boettger et al.'s RECOCASE tool [25] is also comprised of a number of other applications. RECOCASE uses the Link Grammar Parser¹⁷ to parse use-cases and ExtrAns¹⁸ is used to generate “flat logical forms” which are then analysed using FCA.

A paper by Tonella [26] describes the CANTO tool (**C**ode and **A**rchitecture **a**Nalysis **T**ool) [27] which has a modular architecture composed of several subsystems. CANTO consists of a front-end for analysing C code, an architecture recovery tool, a flow analysis tool and a customised editor. The components communicate either via sockets or files and apart from the flow analysis tool each of the components is an external application. Visualisations produced by the architecture recovery tool are created using PROVIS — yet another graph drawing application based on Doty.

Another FCA framework implemented by Arévalo [28,29] and Buchli [30] is ConAn (**C**oncept **A**nalysis) [30]. ConAn is implemented in Smalltalk and consists of a number of tools for the creation and analysis of formal contexts. A tool called *ConAn PaDi* (**C**onAn **P**attern **D**isplay) built using the ConAn framework is used for analysing patterns in data from the *Moose* Smalltalk re-engineering environment¹⁹. Beyond software engineering applications ConAn also represents a generic and extensible framework. Users can provide objects and attributes (known respectively as *elements* and *properties*) as labels in a table or custom Smalltalk objects can be implemented to represent the elements and properties used by ConAn.

The implementation of the authors own tool [31], SpecTrE (the **S**pecification **T**ransformation **E**ngine), mirrors the modular approach taken by Van Deursen and Kuipers. SpecTrE is used for visualising and navigating formal specification documents written in Z [32]. The tool makes use of a parser to extract information from specifications which is stored in a database. The information is then formatted for analysis and visualisation using ToscanaJ and ZML [33] — an XML-based Z representation.

4 Conclusion

This paper has presented an overview of FCA tools ranging from the early DOS-based implementations through to Java-based tools currently under active development. Section 2 introduced the general purpose tools which include both commercial and open-source software. Section 3 then discussed application specific tools. While the modular tools demonstrate the reuse of common components for both the analysis of concepts and visualisation, the diversity represents the applicability of FCA itself to wide-range of problems.

¹⁶ See <http://www.rational.com/products/rose/>

¹⁷ See <http://bobo.link.cs.cmu.edu/link>

¹⁸ See <http://www.ifi.unizh.ch/cl/extrans/overview.html>

¹⁹ See <http://www.iam.unibe.ch/~scg/Research/Moose/>

References

1. Duquenne, V., Chabert, C., Cherfouh, A., Delabar, J.M., Doyen, A.L., Pickering, D.: Structuration of phenotypes/genotypes through galois lattices and implications. In Nguifo, E., Liquière, M., Duquenne, V., eds.: CLKDD'01: Concept Lattices-based Theory, Methods and Tools for Knowledge Discovery in Databases. Volume 42., CEUR (2001) 21–32
2. Burmeister, P.: Formal concept analysis with ConImp: Introduction to the basic features. Technical report, TU-Darmstadt, Darmstadt, Germany (1996)
3. Hereth, J.: DOS programs of the Darmstadt research group on formal concept analysis (1999)
4. Becker, P., Correia, J.H.: The ToscanaJ suite for implementing conceptual info. systems. In Stumme, G., ed.: Proceedings of the First Int'l Conf. on Formal Concept Analysis - ICFCA'03, Springer-Verlag (2003) to appear.
5. Vogt, F., Wille, R.: TOSCANA - a graphical tool for analyzing and exploring data. In Tamassia, R., Tollis, I., eds.: Proceedings of the DIMACS Int'l Workshop on Graph Drawing (GD'94). Lecture Notes in Comp. Sci. 894, Berlin-Heidelberg, Springer-Verlag (1995) 226–233
6. Groh, B.: A Contextual-Logic Framework based on Relational Power Context Families. PhD thesis, Griffith Uni., School of Info. and Communication Tech. (2002)
7. Becker, P.: Multi-dimensional representations of conceptual hierarchies. In: Conceptual Structures—Extracting and Representing Semantics, Contributions to ICCS 2001. (2001) 145–158
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag, Berlin (1999)
9. Valtchev, P., Gosser, D., Roume, C., Hacene, M.: Galicia: an open platform for lattices. In Ganter, B., de Moor, A., eds.: Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag (2003) 241–254
10. Valtchev, P., Missaoui, R., Godin, R., Meridji, M.: Generating frequent itemsets incrementally: two novel approaches based on galois lattice theory. Journal of Experimental and Theoretical Artificial Intelligence (JETAI) : Special Issue on Concept Lattice-based theory, methods and tools for Knowledge Discovery in Databases **14** (2002) 115–142
11. Valtchev, P., Missaoui, R., Godin, R.: A framework for incremental generation of frequent closed itemsets. In: Proceedings of the Workshop on Discrete Mathematics and Data Mining (DM&DM2002), Arlington, VA (2002)
12. Huchard, M., Roume, C., Valtchev, P.: When concepts point at other concepts: the case of UML diagram reconstruction. In: Advances in Formal Concept Analysis for Knowledge Discovery in Databases, FCAKDD 2002. (2002) 32–43
13. Düwel, S.: BASE - ein begriffsbasiertes Analyseverfahren für die Software-Entwicklung. PhD thesis, Philipps-Universität, Marburg (2000)
14. Taran, T., Tkachev, O.: Applications of formal concept analysis in humane studies. In Ganter, B., de Moor, A., eds.: Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag (2003) 271–274
15. Cole, R., Eklund, P.: Scalability in formal concept analysis. Computational Intelligence, **15** (1999) 11–27
16. R. Cole, P.E., Stumme, G.: CEM – a program for visualization and discovery in email. In Zighed, D., Kormorowski, J., Zytow, J., eds.: Proceedings of PKDD 2000. LNAI 1910, Berlin, Springer-Verlag (2000) 367–374
17. Lindig, C., Snelting, G.: Assessing modular structure of legacy code based on mathematical concept analysis. In: Proceedings of the Int'l Conf. on Software Eng. (ICSE 97), Boston (1997) 349–359
18. Snelting, G., Tip, F.: Reengineering class hierarchies using concept analysis. Technical Report RC 21164(94592)24APR97, IBM T.J. Watson Research Center, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA (1997)

19. Lindig, C.: Concept-based component retrieval. In Köhler, J., F., Giunchiglia, Green, C., Walther, C., eds.: Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs. (1995) 21–25
20. van Deursen, A., Kuipers, T.: Identifying objects using cluster and concept analysis. In: Proceedings of the 21st Int'l Conf. on Software Eng., ICSE-99, ACM (1999) 246–255
21. Kuipers, T., Moonen, L.: Types and concept analysis for legacy systems. Technical Report SEN-R0017, Centrum voor Wiskunde en Informatica (2000)
22. Eisenbarth, T., Koschke, R., Simon, D.: Locating features in source code. *IEEE Transactions on Software Eng.* **29** (2003) 195–209
23. Ammons, G., Mandelin, D., Bodik, R., Larus, J.: Debugging temporal specifications with concept analysis. In: Proceedings of the Conf. on Programming Language Design and Implementation PLDI'03, ACM (2003)
24. Bojic, D., Velasevic, D.: Reverse eng. of use case realizations in UML. In: Symposium on Applied Computing - SAC2000, ACM (2000)
25. Böttger, K., Schwitter, R., Richards, D., Aguilera, O., Mollá, D.: Reconciling use cases via controlled language and graphical models. In: INAP'2001 - Proc. of the 14th Int'l Conf. on Applications of Prolog, Japan, Uni. of Tokyo (2001) 20–22
26. Tonella, P.: Concept analysis for module restructuring. *IEEE Transactions on Software Eng.* **27** (2001) 351–363
27. Antoniol, G., Fiutem, R., Lutteri, G., Tonella, P., Zanfei, S.: Program understanding and maintenance with the CANTO environment. In: Proceedings Int'l Conf. on Software Maintenance. (1997) 72–81
28. Arévalo, G.: Understanding behavioral dependencies in class hierarchies using concept analysis. In: Proceedings of LMO 2003 (Langages et Modèles á Object), Paris (France), Hermes (2003)
29. Arévalo, G., Ducass, S., Nierstrasz, O.: Understanding classes using x-ray views. In: MASPEGHI 2003, MAnaging SPEcialization/Generalization Hierarchy (MASPEGHI) Workshop at ASE 2003, Montreal, Canada (2003) Preliminary Version.
30. Buchli, F.: Detecting software patterns using formal concept analysis. Technical Report IAM-03-010, Institut für Informatik und angewandte Mathematik, Universität Bern, Switzerland (2003)
31. Tilley, T.: Towards an FCA based tool for visualising formal specifications. In Ganter, B., de Moor, A., eds.: Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag (2003) 227–240
32. Spivey, J.: The Z notation : a reference manual. Prentice-Hall Int'l (1989)
33. Sun, J., Dong, J., Lui, J., Wang, H.: Object-Z web environment and projections to UML. In: WWW10 - 10th Int'l World Wide Web Conf., New York, ACM (2001) 725–734

Automated Lattice Drawing

Ralph Freese

University of Hawaii, Honolulu, HI 96822, USA

ralph@math.hawaii.edu

<http://www.math.hawaii.edu/~ralph/>

Abstract. Lattice diagrams, known as Hasse diagrams, have played an ever increasing role in lattice theory and fields that use lattices as a tool. Initially regarded with suspicion, they now play an important role in both pure lattice theory and in data representation. Now that lattices can be created by software, it is important to have software that can automatically draw them.

This paper covers:

- The role and history of the diagram.
- What constitutes a good diagram.
- Algorithms to produce good diagrams.

Recent work on software incorporating these algorithms into a drawing program will also be covered.

An *ordered set* $\mathbf{P} = (P, \leq)$ consists of a set P and a partial order relation \leq on P . That is, the relation \leq is reflexive ($x \leq x$), transitive ($x \leq y$ and $y \leq z$ imply $x \leq z$) and antisymmetric ($x \leq y$ and $y \leq x$ imply $x = y$). If P is finite there is a unique smallest relation \prec , known as the *cover* or *neighbor* relation, whose transitive, reflexive closure is \leq . (Graph theorists call this the transitive reduct of \leq .) A *Hasse diagram* of \mathbf{P} is a diagram of the acyclic graph (P, \prec) where the edges are straight line segments and, if $a < b$ in \mathbf{P} , then the vertical coordinate for a is less than the one for b . Because of this second condition arrows are omitted from the edges in the diagram.

A *lattice* is an ordered set in which every pair of elements a and b has a least upper bound, $a \vee b$, and a greatest lower bound, $a \wedge b$, and so also has a Hasse diagram.

These Hasse diagrams¹ are an important tool for researchers in lattice theory and ordered set theory and are now used to visualize data.

This paper deals the special issues involved in such diagrams. It gives several approaches that have been used to automatically draw such diagrams concentrating on a three dimension force algorithm especially adapted for ordered sets that does particularly well.

We begin with some examples.

¹ In the second edition of his famous book on lattice theory [3] Birkhoff says these diagrams are called Hasse diagrams because of Hasse's effective use of them but that they go back at least to H. Vogt, *Résolution algébrique des équation*, Paris, 1895.

1 Some Examples

Generally lattice theorists have a fairly precise idea of how their diagrams should look. Figure 1 gives an example.

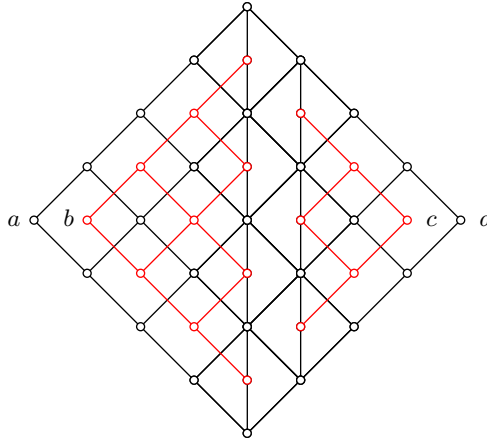


Fig. 1. The diagram of a small modular lattice

Of course the reflection of this diagram around the vertical axis would be considered equally acceptable but Figure 2 is another possible diagram of the same lattice.

While any permutation of the elements of a particular level would produce a correct representation of the order, almost any such permutation would make the diagram unrecognizable.

Most lattice theorists would draw the lattice of subsets of a three element set as the cube drawn on the left of Figure 3. Orthomodular lattice theorists often prefer to draw it as in the center but no lattice theorist would draw it as is done on the right.

Thus while a randomly chosen diagram will be mathematically correct, it is very likely to be useless. Prior to lattices being generated by computer programs this was not a problem: mathematicians knew (at least in most cases) how a particular lattice should be drawn and drew it that way.

2 A Brief History of the Diagram²

As mentioned earlier lattice diagrams are often called Hasse diagrams in honor of H. Hasse who used them effectively. Apparently the earliest lattice theorists

² Pseudo-history might be a better description of this section. The author's mathematical great grand parent is Eric Temple Bell who is known for his colorful, albeit not entirely accurate, accounts of the history of mathematics; see [18].

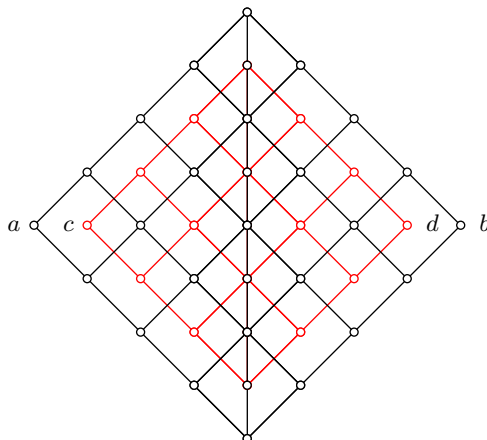


Fig. 2. A different diagram of the lattice of Figure 1

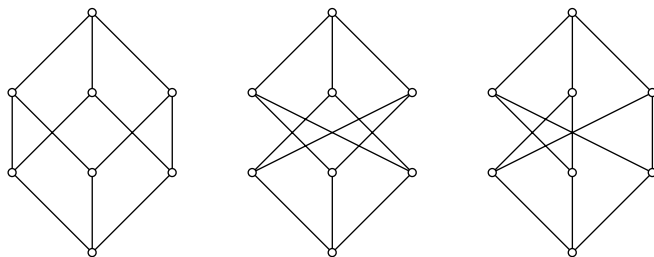


Fig. 3. The lattice of subsets of a three element set. Left: the usual diagram; Center: as sometimes drawn by orthomodular lattice theorists; Right: an ugly version

such as R. Dedekind did not use diagrams to represent lattices. They began to be used in the 1930's but more as a tool for discovering new results; they rarely appeared in the literature.

The free modular lattice with three generators (which is drawn in Figure 8) was first described by Dedekind in [7]. O. Ore [17] gives something resembling a diagram by placing the elements of the free modular on the page in an arrangement reflecting the order but no lines are drawn. Ore, in connection with his work on direct decompositions of algebraic systems, asked R. P. Dilworth if a , b , c and d are elements of a modular lattice such that every pair meets to the least element and all pairs join to the top except possibly the pairs a, b and c, d , do all pairs join to the top? Dilworth drew a diagram to show this was false; see Figure 4. Ore, like many lattice theorists of his era, was skeptical of diagrams and asked Dilworth to find this lattice as a sublattice of the lattice of normal subgroups of a group.

Another interesting thing about this diagram was that Dilworth used both red and black to draw it. (If you are reading a print version of this paper, you can see the color version at the author's web site.) This use of color way back

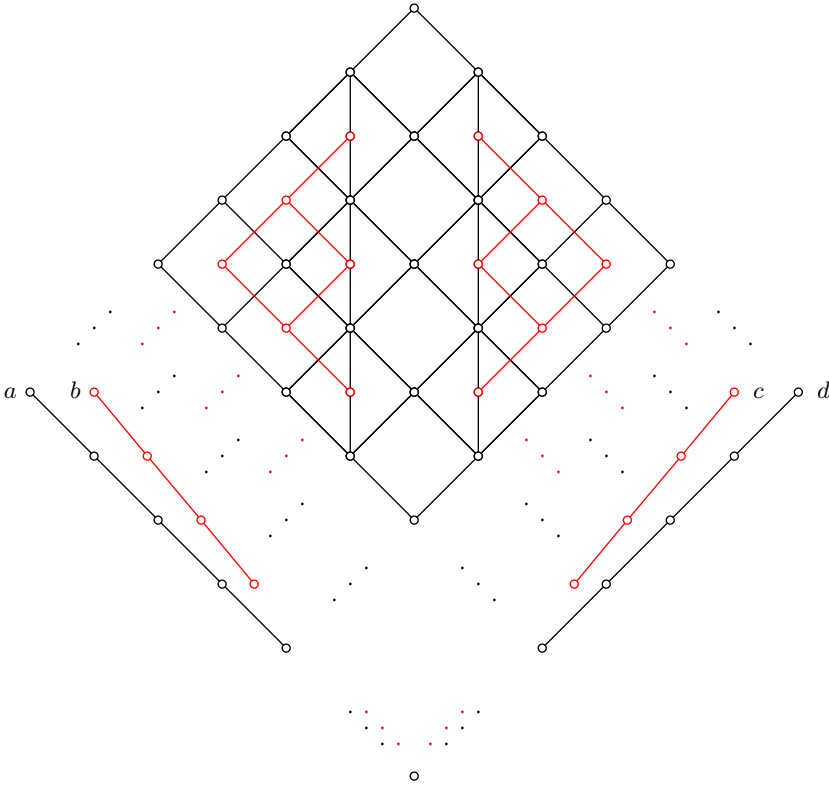


Fig. 4. A counter-example to Ore's question

in the 1930's makes the diagram much easier to understand and better than the diagrams in journals even today.

A. Day, C. Herrmann, and R. Wille [4] constructed a lattice much like this one in their important study of four generated modular lattices. It shows, for example, that \mathbf{M}_4 , the six element lattice with a least and greatest element and 4 incomparable elements, is not a projective modular lattice.

In [8] Dilworth cites as an example J. E. McLaughlin paper [16] in which he proves that an atomic lattice with unique comparable complements must be modular. (See [1] for a history of this important lattice theory problem.) As was often done at that time, McLaughlin used diagrams in finding his proof but omitted them from his paper. Dilworth's paper shows how much more understandable the proof can be with the use of diagrams.

By the 1970's diagrams had become an important tool in lattice theory. The appendix has several diagrams that have played an important role in research, biased toward my own work. Figure 8 shows the free modular lattice on three generators, $\mathbf{FM}(3)$, and the lattice freely generated by \mathbf{M}_3 and two extra ordered elements below one of the generators. Both are used repeatedly in the study of modular lattices.

Figure 10 shows the free modular lattice $\mathbf{FM}(2+2)$ and free lattice $\mathbf{FL}(2+2)$ generated by two 2-element chains. Of course the sublattice generated by two 2-element chains in any lattice must be a homomorphic image of $\mathbf{FL}(2+2)$ and thus these lattices are a useful tool in lattice theory. It is interesting that $\mathbf{FL}(2+2)$ was described by R. A. Dean in [6] but it was H. L. Rolf who diagrammed the lattice in [23] and since then it is known as Rolf's lattice.

The lattice on the left of Figure 11 is J. B. Nation's semidistributive lattice which is not a bounded image of a free lattice (answering a question of R. McKenzie). Nation and I constructed the lattice on the right of Figure 11 from this lattice using Alan Day's doubling construction. We knew this lattice had certain properties and hoped it had certain additional properties but we were having trouble proving that it did. To our surprise we found we were actually able to diagram this lattice. Once we had the diagram of the lattice, it was clear that it had the properties we wanted.

Figure 9 diagrams the congruence lattice of the free algebra on one generator in Polin's variety. This diagram played a critical role in the characterization of varieties with modular congruence lattice of [5].

3 Automatic Drawing

In the early 1980's J. B. Nation and I were working on free lattices. The algorithms for deciding equality of terms and for finding the canonical form of elements were tractable; that is, polynomial time, but rather unpleasant for humans. So we wrote programs to do this (mostly in Lisp). Associated with each element of a free lattice is a finite lattice that determines whether or not the element has lower covers (lower neighbors). For example the lattices associated with $(x \vee (y \wedge z)) \wedge (y \vee z)$ and with $(x \vee (y \wedge z)) \wedge (y \vee (x \wedge z))$ are diagrammed in Figure 5. (Since the second lattice is semidistributive but the first is not, $(x \vee (y \wedge z)) \wedge (y \vee z)$ has a lower cover but $(x \vee (y \wedge z)) \wedge (y \vee (x \wedge z))$ does not; see [10]).

I had programs to automatically calculate these lattices, but it did not give a diagrams. It just gave a list of the elements and covers. It is much more difficult to understand the lattice from this sort of data, just as in Formal Concept Analysis lattices help immensely in visualizing data. So I wrote a program to produce a lattice diagram. But was disappointed because even small, well known lattices were usually unrecognizable. For example the eight element Boolean algebra, drawn on the left of Figure 3 often looked like the center or right drawing. Clearly something better was needed.

4 Our Algorithm

Our algorithm is based on a combination of a mathematical rank function to determine the height and a modification of the "forces" method of graph theory.

The program first calculates the rank function on the ordered set and uses this to determine the height of the elements. It then places the points in three

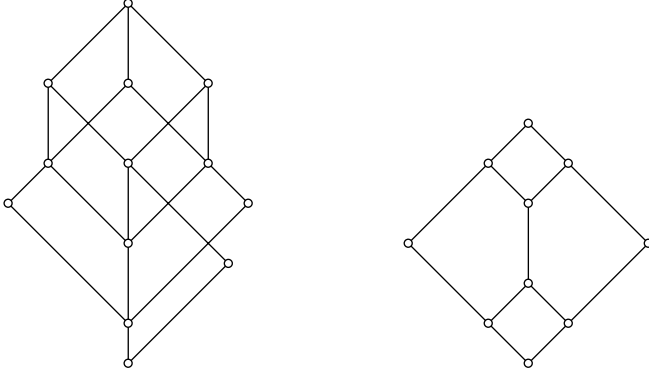


Fig. 5. The lattices $L((x \vee (y \wedge z)) \wedge (y \vee z))$ and $L((x \vee (y \wedge z)) \wedge (y \vee (x \wedge z)))$

space using the rank for the height, i.e., for the z -coordinate. The points of the same rank are arranged around a circle on a plane parallel to the x - y plane. We now place imaginary forces on the elements. Comparable elements are attracted to each other while incomparable elements are repulsed.

These forces are applied several times in three phases. In the first phase the repulsive force is set to be very strong; in the second phase the attractive force is strong; and in the final phase the forces are balanced. Then an optimal projection to two space is chosen (either by the program or the user).

4.1 The Rank Function

As mentioned above the algorithm positions the elements of the ordered set \mathbf{P} in 3-space and uses forces to adjust the x and y coordinates but the z coordinate (the height) is determined by a rank function on the ordered set given by

$$\text{rank}(a) = \text{height}(a) - \text{depth}(a) + M$$

where $\text{height}(a)$ is the length of the longest chain from a to a minimal element, $\text{depth}(a)$ is the length of the longest chain from a to a maximal element. M can be any constant; it is usually the length of the longest chain in \mathbf{P} so that the rank of the least element of a lattice is 0. Figure 6 gives an example of this rank function. Notice the rank function has a nice top to bottom symmetry.

The usual algorithm to find a linear extension of \mathbf{P} (graph theorists call this *topological sorting*) can be easily modified to calculate this rank function in linear time. (See Chapter 11 of [10] for a discussion of algorithms for ordered sets and lattices.)

4.2 Initialization and Force Scheme

We associate a point (x, y, z) in 3-space with each element of the ordered set. The z coordinate is determined by the rank function. Initially the points of the

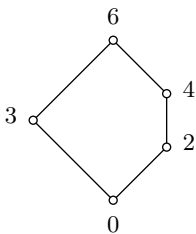


Fig. 6. The rank function for N_5

same rank are arranged with equal spacing (actually a slight perturbation is added to avoid unstable equilibria) around a circle on a plane parallel to the x - y plane with radius equal to the number of elements of that rank.

Then forces (with heavy damping—imagine the points lying in a thick syrup) are repeatedly applied to the points. Each point is attracted to the points it is comparable with. If the point has coordinates (x_0, y_0, z_0) and is comparable with a point with coordinates (x_1, y_1, z_1) the force on the first point is

$$c_{\text{att}} \langle x_1 - x_0, y_1 - y_0, 0 \rangle$$

If these two points are incomparable they are repulsed. The force on the first point is

$$\frac{c_{\text{rep}} \langle x_0 - x_1, y_0 - y_1, 0 \rangle}{|(x_1 - x_0)|^3 + |(y_1 - y_0)|^3 + |(z_1 - z_0)|^3}$$

Note the attraction of two comparable elements does not depend on the z -coordinate while the repulsion does. In both cases the force is in the x - y plane so that the z -coordinate remains fixed. The attraction is similar to the attraction of a spring (Hooke's Law) with natural length 0. But since the z -coordinate is unchanged the distance between the two points is always at least $|z_1 - z_0|$. Also note that the attraction is not just between elements and their upper and lower covers (neighbors) but between all pairs of comparable elements and does not depend on $z_1 - z_0$. The repulsion force uses an inverse square law and it does depend on the z -coordinate.

4.3 Iteration

The total force on each node is calculated and all of the nodes are moved. This process is iterated until the diagram becomes stable. Of course the attraction and repulsion can be modified by changing c_{att} and c_{rep} and the current algorithm goes through three stages: first with the repulsion force strong (that is c_{rep} large), then with the attraction force strong and finally with balanced forces.

Finally a projection of the form $\langle x, y, z \rangle \mapsto \langle x \cos \theta + y \sin \theta, z \rangle$ into the plane is chosen. θ can be chosen to maximize some niceness criterion or the user can rotate the picture to find a pleasing projection.

4.4 Comparisons

While the results of this algorithm do not match the quality of hand drawn diagrams they are at least recognizably the same and often close to being aesthetically pleasing. For lattices with 10 or less elements the result is usually very close to the hand drawn version.

To illustrate the difference we have taken three figures (Figure 1.1, Figure 1.17, and Figure 1.25) from [11], and reproduced them along with the diagram drawn by the program in Figures 12, 13, and 14. The first two, while far from perfect, are easily recognized. The third one, Figure 14 is particularly interesting because the computer generated diagram is actually the way it is usually drawn; see, for example, Figure 1 of Knuth [15]. It better shows off the symmetry of the lattice. The diagram from [11] also has some obvious pluses.

5 What Is a ‘Nice’ Diagram?

There is a large body of work on graph drawing. The Hasse diagram is a very restricted type of graph diagram: the edges must be straight lines and vertical coordinate must agree with the order (greater elements of the ordered set must be higher in the diagram). Since our goal is to really picture the lattice, we rarely diagram lattices with more than 50 elements so asymptotic results are not always relevant. Nevertheless there are results of interest to lattice drawing. We discuss one: minimizing the number of edge crossings.

5.1 Edge Crossings

Finding a diagram with the minimum number of edge crossings is NP-hard as was shown by Garey and Johnson [13]. In [9] P. Eades and N. Wormald showed that the problem of minimizing the edge crossings in a bipartite graph where one of the layers has a prescribed order is also NP-hard. It is easy to see that these results imply that the problem of minimizing crossings in an ordered set is NP-hard. However this does not immediately imply that the crossing problem for lattices is hard³. It was Ivan Rival who brought the crossing problem for lattice to my attention. While the result of Garey and Johnson can be modified in a straight-forward way to apply to lattices, extending Eades and Wormald’s result takes more care. We outline how to do it here. Stating the problem:

DECISION CROSSING PROBLEM FOR LATTICES (DCPL)

Instance: A lattice \mathbf{L} , an ordering on the atoms of \mathbf{L} and an integer M .

Question: Is there a diagram of \mathbf{L} , that is, a map from the elements of \mathbf{L} into the plane, such that the vertical coordinates are given by the rank function and the left to right order of the atoms determined by the horizontal coordinate is the given ordering of the atoms and such that the number of crossings is at most M ?

³ Satisfying the lattice axioms places strong restrictions on a ordered set. While the diagram of an ordered set with n elements can have as many as $n^2/4$ edges, that of a lattice can have at most $n^{3/2}$; see [10].

Theorem 1. DCPL is NP-complete.

Proof. The problem is clearly in the class NP. The proof that it is NP-complete follows the proof of Theorem 1 of [9] so we will only point out how to modify that theorem to prove this theorem. To prove this problem is NP-complete we give a polynomial-time reduction to the following known NP-complete problem: given a directed graph \mathbf{D} and a positive integer K is there a set of at most K arcs whose removal would make \mathbf{D} acyclic? This is known as the Feedback Arc Set problem and is NP-complete; see p. 192 of [12].

So let $\mathbf{D} = (U, B)$ be a directed graph with vertex set $U = \{u_1, \dots, u_n\}$ and arc set B and let K be a positive integer. For each arc $a = (u_r, u_s) \in B$ let $C(a)$, let the “clump” associated with a be the set

$$C(a) = \{c_1^a, c_2^a, c_5^a, c_6^a\} \cup \{c_{3,i}^a, c_{4,i}^a : 1 \leq i \leq n, i \neq r, s\}.$$

The atoms of \mathbf{L} consist of the union of the $C(a)$ ’s. The left to right order of these atoms keeps the elements of each clump together and within a clump the order is the order on the first index and within the $c_{3,i}^a$ ’s on the second index but for the $c_{4,i}^a$ ’s it is the *reverse* order of the second index.

The coatoms of \mathbf{L} are the elements of U . If $a = (u_r, u_s) \in B$ then in \mathbf{L}

$$\begin{aligned} c_1^a, c_5^a &< u_r \\ c_2^a, c_6^a &< u_s \\ c_{3,t}^a, c_{4,t}^a &< u_t \quad \text{for } t \neq r, s \end{aligned}$$

Since each atom has exactly one upper cover under this order, \mathbf{L} is a lattice once we add a least and greatest element.

Let $x(u_i)$ be the x -coordinate of u_i in a Hasse diagram of \mathbf{L} . This determines a (left to right) order on the coatoms since the rank of all the coatoms is the same. Let B' denote $\{(u_i, u_j) \in B : x(u_i) > x(u_j)\}$. Letting $\beta = |B'|$, the number of crossings in the diagram is

$$4 \binom{\beta}{2} \binom{n}{2} + 2\beta \binom{n-2}{2} + 4\beta(n-1) + \beta + 2|B'|$$

The proof of this is similar to the proof of Lemma 1 of [9]. For example, if $a = (u_r, u_s)$ and $\{i, j\} \cap \{r, s\} = \emptyset$ then, because of the reverse order of the $c_{4,k}^a$ ’s, there there are 2 crossings of the 4 edges from u_i and u_j into $C(a)$ regardless of whether $i < j$ or not. These crossings give rise to the $2\beta \binom{n-2}{2}$ term.

Of course the graph $(U, B - B')$ has a linear extension and so is acyclic. Thus there is a set of K arcs of \mathbf{D} whose removal makes it acyclic if and only there is a left to right ordering of the coatoms of \mathbf{L} so that the diagram of \mathbf{L} with this layout of the coatoms has at most

$$4 \binom{\beta}{2} \binom{n}{2} + 2\beta \binom{n-2}{2} + 4\beta(n-1) + \beta + 2K$$

crossings. Thus DCPL is NP-complete.

5.2 Heuristic Niceness

Of course I am more interested in a useful tool than in theoretical results about diagrams (which is also the goal of FCA) and so tend to concentrate on heuristics. What constitutes a “nice” diagram is, of course, a question of æthetics, not mathematics. R. Wille and his colleagues at Darmstadt suggested various criteria. One was minimizing the number of different slopes of the edges. Looking at the diagrams in this paper we see that this is very important. But it is interesting to note that some minimum slope diagrams can be bizarre. Figure 7 from [25] shows an Escher-like diagram of the 8-element Boolean algebra.

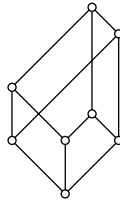


Fig. 7. An Escher-like diagram of the eight element Boolean algebra

In [11] this idea is refined into two rules: the *rule of parallelograms* and the *rule of lines*. While the algorithm we presented does not follow these rules, it often produces diagrams that approximate them.

In [2] the authors present an algorithm for minimizing total length of the edges when the elements are restricted to lie on a grid which produces nice results.

6 Software

Our algorithm was originally implemented in Lisp, but in 1996 we converted it to a Java applet; see <http://www.math.hawaii.edu/~ralph/LatDraw/>. It has been used as a component in several other programs:

- JavaMath, a web based application using the applet to draw lattice (particularly subgroup lattices) directly from Maple and GAP.
- Universal Algebra Calculator, an application for universal algebra incorporating our program to draw congruence lattices.
- JaLaBA, Java Lattice Building Application. This allows you to enter your own formal context and it will generate the lattice.

There are more details on the site above.

We are presently reconstituting our program as a component which can be easily plugged into other programs. The goals of the new program include:

- Bring the Java up-to-date: the original applet was written in the first Java (version 1.02) and as such uses the old event model and does not use Graphics2D.

- Add new, often requested, features such as dragging the elements, interval viewing, saving in various formats, printing.
- Two way communication between the component and the client program using it: the program would be aware of events in the component so, for example, it could display information about a node when the user clicks it. In the other direction, the program can request that the component change the color of certain elements in the lattice based on a user's gesture in the program. At the programming level, there will be higher level events so, for example, the program will be able to listen for higher events such as "vertex clicked" or "edge clicked" rather than just mouse clicked.

I plan is to make this open source and to put it on sourceforge.net. Sample programs using the drawing component will be included.

References

1. M. E. Adams, *Uniquely complemented lattices*, The Dilworth Theorems, Selected Papers of Robert P. Dilworth (K. Bogart, R. Freese, and J. Kung, eds.), Birkhäuser, Basel, 1990, pp. 79–84.
2. A. Aeschlimann and J. Schmid, *Drawing orders using less ink*, Order **9** (1992), 5–13.
3. G. Birkhoff, *Lattice theory*, Amer. Math. Soc., Providence, R. I., 1948, rev. ed., Colloquium Publications.
4. A. Day, Ch. Herrmann, and R. Wille, *On modular lattices with four generators*, Algebra Universalis **2** (1972), 317–323.
5. Alan Day and Ralph Freese, *A characterization of identities implying congruence modularity, I*, Canad. J. Math. **32** (1980), 1140–1167.
6. R. A. Dean, *Completely free lattices generated by partially ordered sets*, Trans. Amer. Math. Soc. **83** (1956), 238–249.
7. R. Dedekind, *Über die von drei Moduln erzeugte Dualgruppe*, Math. Annalen **53** (1900), 371–403.
8. R. P. Dilworth, *The role of order in lattice theory*, Ordered Sets (I. Rival, ed.), D. Reidel, Dordrecht, Holland, 1982, Proc. of the Banff Symposium on Ordered Sets.
9. P. Eades and N. Wormald, *Edge crossings in drawings of bipartite graphs*, Algorithmica **11** (1994), no. 4, 379–403.
10. Ralph Freese, Jaroslav Ježek, and J. B. Nation, *Free lattices*, Amer. Math. Soc., Providence, 1995, Mathematical Surveys and Monographs, vol. 42.
11. B. Ganter and R. Wille, *Formal Concept Analysis*, Springer, Berlin, Heidelberg, New York, 1999, ISBN 3-540-62771-5.
12. M. Garey and D. Johnson, *Computers and intractability, a guide to the theory of NP-completeness*, W. H. Freeman and Company, San Francisco, 1979.
13. M. R. Garey and D. S. Johnson, *Crossing number is NP-complete*, SIAM J. Algebraic Discrete Methods **4** (1983), no. 3, 312–316.
14. A. Garg and R. Tamassia, *Advances in graph drawing*, Algorithms and Complexity, Springer Verlag, Berlin-New York, 1994, Second Italian Conference, CASC '94, Rome Italy, pp. 12–21.
15. Donald E. Knuth, *The art of computer programming. Vol. 3*, second ed., Addison-Wesley Publishing Co., Reading, Mass., 1998, Searching and Sorting, Addison-Wesley Series in Computer Science and Information Processing.

16. J. E. McLaughlin, *Atomic lattices with unique comparable complements*, Proc. Amer. Math. Soc. **7** (1956), 864–866.
17. O. Ore, *On the foundations of abstract algebra, I*, Ann. Math. **36** (1935), 406–437.
18. Constance Reid, *The search for E. T. Bell: also known as John Taine*, Math. Assoc. Amer., Washington, D. C., 1993.
19. I. Rival, *The diagram*, Graphs and order, NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci., vol. 147, Reidel, Dordrecht, Boston, 1985, pp. 103–133.
20. I. Rival, *Reading, drawing, and order*, Algebras and orders, NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., vol. 389, Kluwer Acad. Publ., Dordrecht, 1991, pp. 359–404.
21. I. Rival and B. Sands, *Pictures in lattice theory*, Algebraic and geometric combinatorics **65** (1982), 341–355.
22. I. Rival and R. Wille, *Lattices freely generated by partially ordered sets: which can be “drawn”?*, J. Reine Angew. Math. **310** (1979), 56–80.
23. H. L. Rolf, *The free lattice generated by a set of chains*, Pacific J. Math. **8** (1958), 585–595.
24. M. Skorsky, *Endliche Verbände—Diagramme und Eigenschaften*, Ph.D. thesis, Technische Hochschule Darmstadt, 1992.
25. J. Stephan, *Liniendiagramme von verbänden*, Technische Hochschule Darmstadt, Darmstadt, 1987, Diplomarbeit.
26. R. Wille, *Lattices in data analysis: how to draw them with a computer*, Algorithms and order (I. Rival, ed.), Univ. Ottawa, 1989, pp. 33–58.

Appendix A: More Diagrams

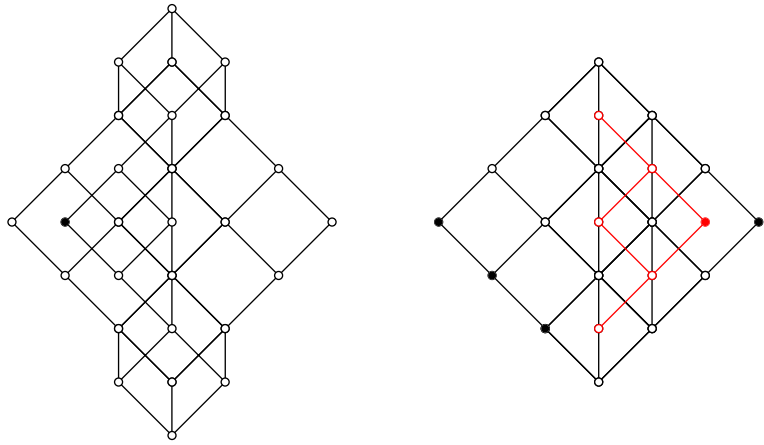


Fig. 8. $\mathbf{FM}(3)$ and the lattice freely generated by \mathbf{M}_3 and two extra ordered elements below a generator

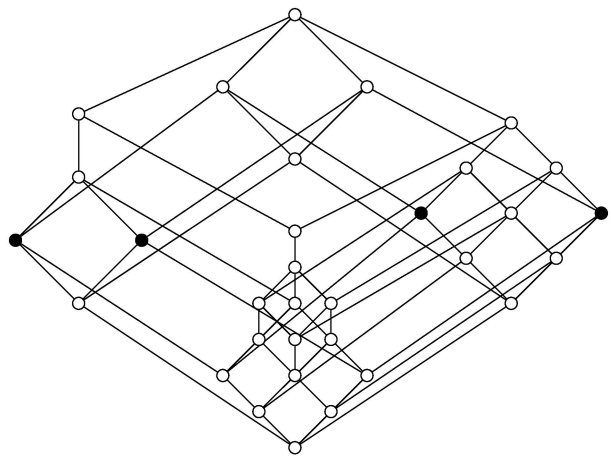


Fig. 9. The congruence lattice of the free algebra on one generator in Polin's variety

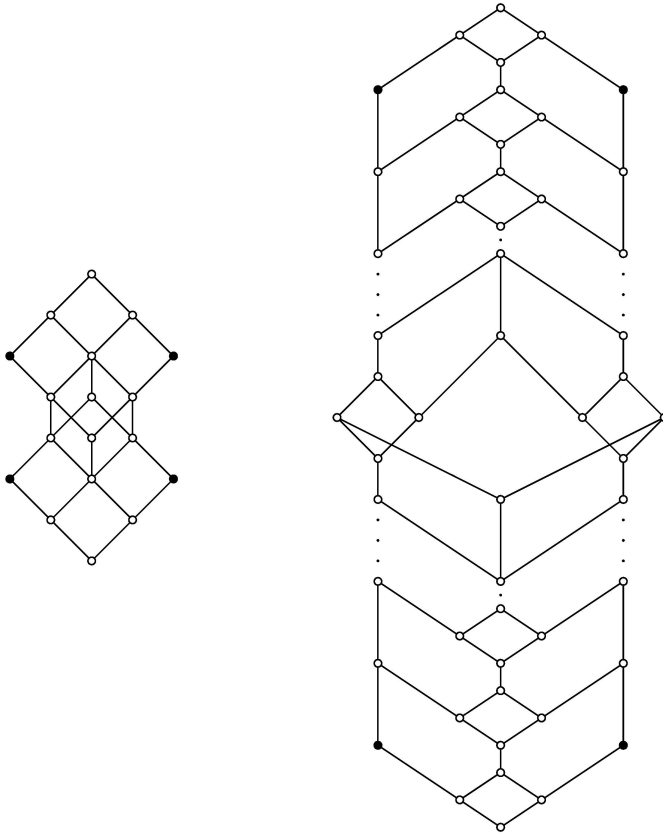


Fig. 10. $\mathbf{FM}(2 + 2)$ and $\mathbf{FL}(2 + 2)$

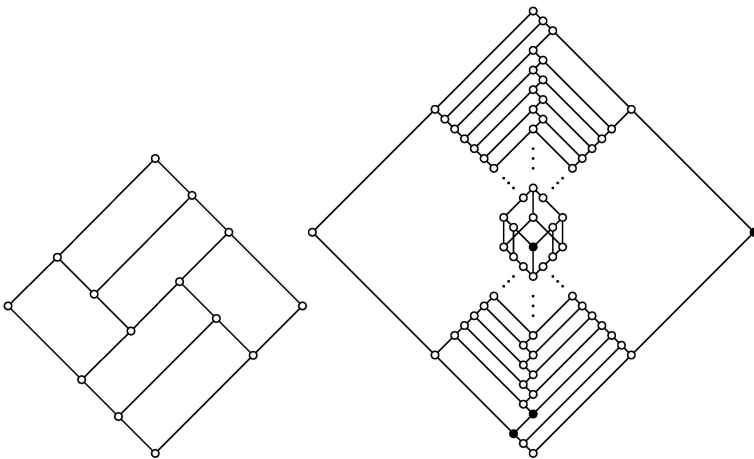


Fig. 11. Nation's example of a finite semidistributive lattice which is not a bounded homomorphic image of a free lattice and its W -cover

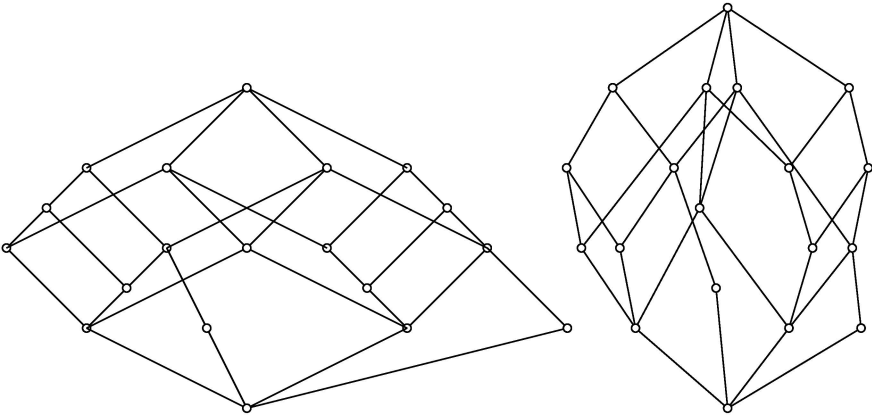


Fig. 12. Left: Figure 1.1 of [11]. Right: as drawn by the program

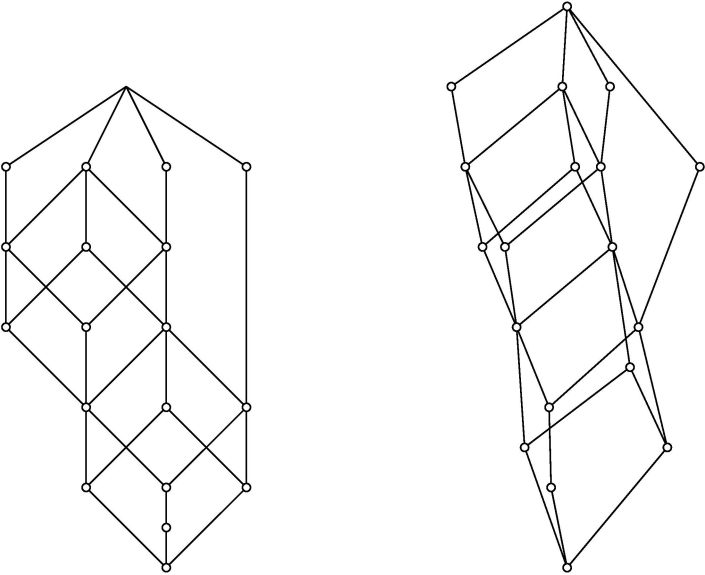


Fig. 13. Left: Figure 1.17 of [11]. Right: as drawn by the program

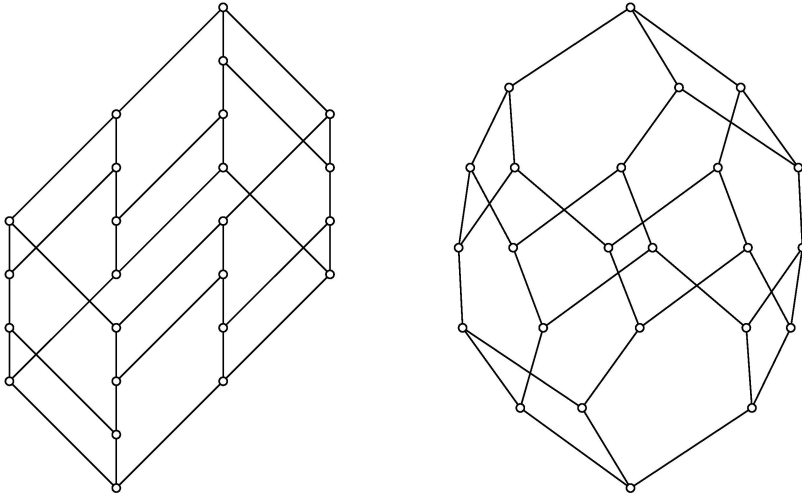


Fig. 14. Left: Figure 1.25 of [11]. Right: as drawn by the program

Congruences of Finite Distributive Concept Algebras

Bernhard Ganter

Institut für Algebra, Technische Universität Dresden, Germany

Abstract. We describe a method to construct a formal context for the congruence lattice of a finite distributive concept algebra. This is part of a broad effort to investigate the structural properties of conceptual negations.

1 Introduction

“...these animals have four legs, but they aren’t mammals. They are not dangerous, they don’t attack humans, but some of them have very poisonous skin...”

This could be a partial natural language description of some concept, perhaps of “frog”. Note that it contains several negations, even iterated ones like “some”, which perhaps can be interpreted by an $(\neg\forall\neg)$ -expression. Formal Concept Analysis and Contextual Logic should be able to work with such expressions, not only on the context level, but also on the algebraic level of concept lattices.

It seems that there are many possibilities to formalize conceptual negation. Rudolf Wille has therefore started a broad investigation of negation notions, their formal representations, and their algebraic properties. This paper deals with one small aspect of this research. It gives an impression how gradually mathematical theory is accumulated that eventually should be useful in practical applications. But our approach is not guided by potential applications. Our aim is to prove a mathematical result, which then hopefully is a building block in a powerful structure theory.

2 Concept Algebras

The concept lattice $\mathbf{V} := \mathfrak{B}(G, M, I)$ of any formal context (G, M, I) can naturally be equipped with two unary operations

$$\triangle : \mathbf{V} \rightarrow \mathbf{V} \quad \text{and} \quad \nabla : \mathbf{V} \rightarrow \mathbf{V},$$

called **weak negation** and **weak opposition**, given by

$$\begin{aligned} (A, B)^\triangle &:= ((G \setminus A)'', (G \setminus A)') \\ (A, B)^\nabla &:= ((M \setminus B)', (M \setminus B)''). \end{aligned}$$

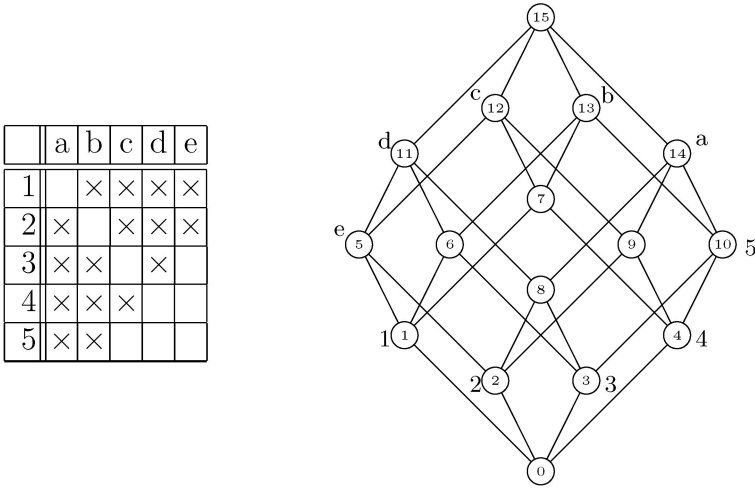


Fig. 1. A formal context and its concept lattice.

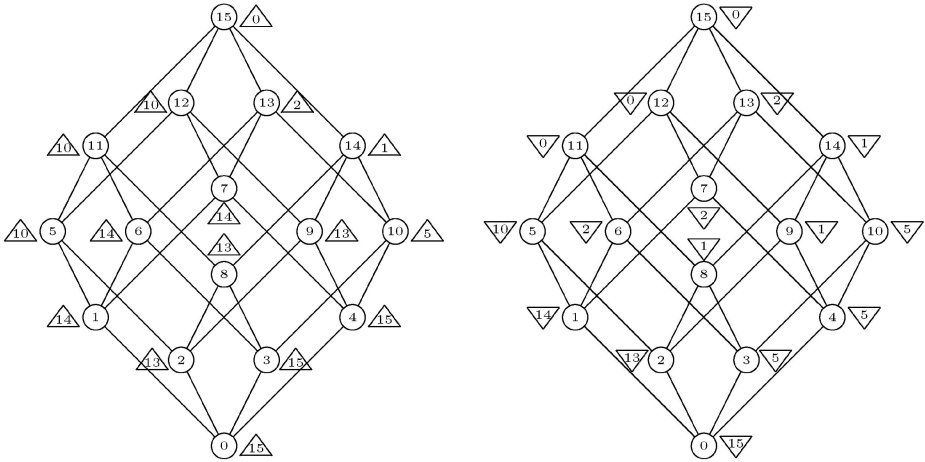


Fig. 2. The dicomplementation of the concept algebra $\underline{\mathfrak{A}}(G, M, I)$, where (G, M, I) is the formal context from Figure 1. The number in the triangle next to a concept gives its weak negation and its weak opposition, respectively.

The concept lattice, together with this natural **dicomplementation**, is called the **concept algebra** $\underline{\mathfrak{A}}(G, M, I)$. A simple example is given in Figures 1 and 2.

This notion was introduced by R. Wille [4], who also started an investigation of the abstract properties of concept algebras. His research was continued and extended by L. Kwuida in his recent Ph.D. project. One of the key questions is to describe the equational or quasi-equational theory of concept algebras and to thereby characterize **dicomplemented lattices** internally. It is, however, still

an unsolved problem to give a compact algebraic description of this class. Only a few elementary axioms are known to hold:

- | | |
|---|--|
| (1) $x^{\Delta\Delta} \leq x$, | (1') $x^{\nabla\nabla} \geq x$, |
| (2) $x \leq y \Rightarrow x^{\Delta} \geq y^{\Delta}$, | (2') $x \leq y \Rightarrow x^{\nabla} \geq y^{\nabla}$, |
| (3) $(x \wedge y) \vee (x \wedge y^{\Delta}) = x$, | (3') $(x \vee y) \wedge (x \vee y^{\nabla}) = x$. |

It has recently been shown by Kwuida [3] that in the case of finite distributive lattices these conditions indeed characterize concept algebras up to isomorphism.

Note that a lattice usually admits more than one, often many dicomplementations. Consider, for example, the Boolean lattice on n atoms. A natural dicomplementation is given by taking the usual Boolean complement for both Δ and ∇ . But this is not the only choice. Kwuida has shown that the total number of dicomplementations on this lattice is in fact $2^{n \cdot (n-1)}$. This comes from the fact that different formal contexts may have isomorphic concept lattices, but non-isomorphic concept algebras. Reducing a context can change the induced dicomplementation. It does therefore not suffice, not even for finite concept algebras $\underline{\mathfrak{A}}(G, M, I)$, to consider only the standard context given by the sets G_{irr} and M_{irr} of irreducible objects and attributes.

3 Congruences of Concept Lattices

Concept algebras do not form an equational class, since concept lattices do not. We nevertheless may expect that complete subalgebras, products and complete homomorphic images of concept algebras are again (isomorphic to) concept algebras. For subalgebras and products, this has been shown by Kwuida, for homomorphic images this is still open. It is therefore of interest to study the homomorphic images of concept algebras, and the congruence relations they induce. This is particularly promising because the congruence theory of concept lattices is so elegant and effective, at least in the case of doubly founded lattices¹. We give a short sketch in the sequel. The reader is referred to [1], Chapter 3, for further basic facts and notions.

- A subcontext $(H, N, I \cap H \times N)$ of (G, M, I) is called **compatible** iff for each concept $(A, B) \in \underline{\mathfrak{B}}(G, M, I)$ the restriction

$$(A \cap H, B \cap N)$$

is a concept of the subcontext.

- If $(H, N, I \cap H \times N)$ is compatible, then the map

$$(A, B) \xrightarrow{\varphi_{H,N}} (A \cap H, B \cap N)$$

automatically is a complete homomorphism

$$\varphi_{H,N} : \underline{\mathfrak{B}}(G, M, I) \rightarrow \underline{\mathfrak{B}}(H, N, I \cap H \times N)$$

between the concept lattices.

¹ The property of being **doubly founded** is a generalization of finiteness [1].

- If $\underline{\mathfrak{B}}(G, M, I)$ is doubly founded, then for each complete homomorphism

$$\varphi : \underline{\mathfrak{B}}(G, M, I) \rightarrow \mathbf{V}$$

from $\underline{\mathfrak{B}}(G, M, I)$ to an arbitrary complete lattice \mathbf{V} there is a compatible subcontext $(H, N, I \cap H \times N)$ with $H \subseteq G_{\text{irr}}$, $N \subseteq M_{\text{irr}}$, and a monomorphism

$$\varepsilon : \underline{\mathfrak{B}}(H, N, I \cap H \times N) \rightarrow \mathbf{V},$$

such that

$$\varphi = \varepsilon \cdot \varphi_{H,N}.$$

- Moreover, the congruence

$$\Theta_{H,N} := \ker \varphi_{H,N} = \ker \varphi$$

is given by

$$\begin{aligned} (A, B)\Theta_{H,N}(C, D) &\iff B \cap N = D \cap N \\ &\iff A \cap H = C \cap H. \end{aligned}$$

- Compatible subcontexts can nicely be described by means of the **arrow relations**. For a formal context (G, M, I) , for $g \in G$ and $m \in M$, we define

$$\begin{aligned} g \swarrow m &: \iff (g, m) \notin I \text{ and } (g' \subseteq h', g' \neq h' \text{ implies } hIm), \\ g \nearrow m &: \iff (g, m) \notin I \text{ and } (m' \subseteq n', m' \neq n' \text{ implies } gIn), \\ g \nearrow m &: \iff g \swarrow m \text{ and } g \nearrow m. \end{aligned}$$

- A subcontext $(H, N, I \cap H \times N)$ is **arrow-closed** if

$$\begin{aligned} h \in H, m \in M, h \nearrow m &\Rightarrow m \in N, \text{ and} \\ g \in G, n \in N, g \swarrow n &\Rightarrow g \in H. \end{aligned}$$

In a doubly founded context, the compatible subcontexts are precisely the arrow closed subcontexts.

- For a doubly founded context (G, M, I) , define

$$g \not\ll m : \iff \begin{cases} \text{there are } g_1 = g, g_2, \dots, g_k \in G \text{ and} \\ m_1, \dots, m_{k-1}, m_k = m \in M \\ \text{such that } g_i \swarrow m_i \text{ holds for all } i \in \{1, \dots, k\} \\ \text{and } g_i \nearrow m_{i-1} \text{ holds for all } i \in \{2, \dots, k\} \end{cases}$$

$$g \not\ll m : \iff \text{not } g \not\ll m.$$

Then a subcontext $(H, N, I \cap H \times N)$ is compatible if and only if $(G \setminus H, N)$ is a formal concept of $\underline{\mathfrak{B}}(G, M, \not\ll)$.

- Therefore the congruence lattice of $\underline{\mathfrak{B}}(G, M, I)$ is isomorphic to the completely distributive concept lattice $\underline{\mathfrak{B}}(G, M, \not\ll)$.

4 Sublattices and Closed Relations

The complete congruences of a concept algebra $\underline{\mathfrak{A}}(G, M, I)$ are those congruences of the concept lattice $\underline{\mathfrak{B}}(G, M, I)$, that preserve the dicomplementation. It follows from general algebraic facts that such congruences form a complete sublattice of the congruence lattice. Formal Concept Analysis offers a nice characterization of the complete sublattices using *closed subrelations*:

- A relation $J \subseteq I$ is a **closed subrelation** of (G, M, I) iff every formal concept of (G, M, J) is also a formal concept of (G, M, I) .
- If J is a closed subrelation, then $\underline{\mathfrak{B}}(G, M, J)$ is a complete sublattice of $\underline{\mathfrak{B}}(G, M, I)$. Conversely, each complete sublattice corresponds to some closed subrelation.
- Therefore: The concept algebra congruences of $\underline{\mathfrak{A}}(G, M, I)$ are in 1-1-correspondence to the concepts of some closed subrelation of $(G_{\text{irr}}, M_{\text{irr}}, \not\equiv)$.

We now know what to look for: a certain subrelation of $\not\equiv$. Before we proceed, we deviate for an intermezzo, studying closed relations of such contexts in greater generality.

4.1 Quasi-ordered Contexts

Let (G, M, I) be a formal context and let \equiv be a subrelation of I such that for each $g \in G$ there is some $m \in M$ with $g \equiv m$, and for each $m \in M$ there is some $g \in G$ with $g \equiv m$.

Slightly misusing notation, we will sometimes write $m \equiv g$ instead of $g \equiv m$, and moreover will combine several relational expressions into single terms. For example, “ $g I m \equiv h$ ” is short for “ $g I m$ and $h \equiv m$ ”.

Definition 1 We call (G, M, I) **quasi-ordered** (over \equiv) if

$$g_1 I m_1 \equiv g_2 I m_2 \text{ implies } g_1 I m_2. \quad \diamond$$

For example, the **ordinal scale** (P, P, \leq) is quasi-ordered over the equality relation $=$.

The reason for calling such a context *quasi-ordered* is the following: We can define relations \leq_M on M and \leq_G on G by

$$\begin{aligned} g_1 \leq_G g_2 &: \iff g_2 I m_1 \equiv g_1 && \text{for some } m_1 \in M \\ m_1 \leq_M m_2 &: \iff m_2 \equiv g_2 I m_1 && \text{for some } g_2 \in G. \end{aligned}$$

Proposition 1 If (G, M, I) is quasi-ordered, then both \leq_G and \leq_M are quasi-orders.

Proof It suffices to give a proof for \leq_G . We have to show that \leq_G is reflexive and transitive.

Reflexivity: For each $g \in G$ there is some $m \in M$ with $g \equiv m$, which implies $g I m$. Thus we get $g I m \equiv g$, and consequently $g \leq_G g$.

Transitivity: Suppose $g_3 \leq_G g_2 \leq_G g_1$. Then

$$g_2 I m_3 \equiv g_3 \text{ for some } m_3 \in M \quad \text{and} \quad g_1 I m_2 \equiv g_2 \text{ for some } m_2 \in M,$$

which combines to

$$g_1 I m_2 \equiv g_2 I m_3.$$

Since (G, M, I) is quasi-ordered, this implies $g_1 I m_3 \equiv g_3$, which is

$$g_3 \leq_G g_1.$$

□

From each of these two quasi-orders we get an equivalence relation by defining

$$\begin{aligned} g_1 \sim_G g_2 &: \iff g_1 \leq_G g_2 \text{ and } g_2 \leq_G g_1, \\ m_1 \sim_M m_2 &: \iff m_1 \leq_M m_2 \text{ and } m_2 \leq_M m_1. \end{aligned}$$

Proposition 2

- $g_1 \equiv m \equiv g_2$ implies $g_1 \sim_G g_2$, and
- $m_1 \equiv g \equiv m_2$ implies $m_1 \sim_M m_2$.

Proof Since \equiv is a subrelation of I , $g_1 \equiv m \equiv g_2$ implies $g_1 I m \equiv g_2$, which is $g_2 \leq_G g_1$. The rest follows analogously. □

Proposition 3 $g_1 \equiv m_1 \leq_M m_2 \equiv g_2$ implies $g_1 \leq_G g_2$, and dually.

Proof If $m_1 \leq_M m_2$, then there is some $g \in G$ with $m_2 \equiv g I m_1$. From $g I m_1 \equiv g_1$ we get $g_1 \leq_G g$ and from $g \equiv m_2 \equiv g_2$ we get $g \leq_G g_2$. □

Proposition 4 If $g_1 \leq_G g_2$ and $g_1 I m$, then $g_2 I m$. Dually, if $m_1 \leq_M m_2$ and $g I m_2$, then $g I m_1$.

Proof $g_1 \leq_G g_2$ implies that for some $m_1 \in M$ we have $g_2 I m_1 \equiv g_1$ and therefore

$$g_2 I m_1 \equiv g_1 I m,$$

which implies $g_2 I m$. □

It is now apparent what the structure of a quasi-ordered context is: After clarification, it is isomorphic to a context (P, P, \leq) for some ordered set (P, \leq) . It is therefore not surprising that we can characterize the formal concepts of the complementary context:

Proposition 5 Let (G, M, I) be a quasi-ordered context (over \equiv). Then (A, B) is a concept of the complementary context $(G, M, (G \times M) \setminus I)$ if and only if the following conditions are fulfilled:

- A is an order ideal of \leq_G ,
- B is an order filter of \leq_M ,
- $A = B^\neq$, and $B = A^\neq$.

Proof First suppose that (A, B) satisfies the conditions of the proposition. If $m \in M$ is not in A' (where A' is computed with respect to the formal context $(G, M, (G \times M) \setminus I)$), then $a I m$ for some $a \in A$. We find some $g \in G$ with $g \equiv m$, which gives $a I m \equiv g$, thus $g \leq_G a$. Since A is an order ideal, this yields $g \in A$ and therefore $m \notin A^\neq = B$. This proves $B \subseteq A'$. For the other inclusion, let $m \notin B$. Then, since $B = A^\neq$, we find some $a \in A$ with $a \equiv m$ and thus $a I m$, which shows that $m \notin A'$.

Conversely, let (A, B) be a formal concept of $(G, M, (G \times M) \setminus I)$. Then

$$B = \{m \in M \mid \neg(a I m) \text{ for all } a \in A\}.$$

According to Proposition 4, $m_1 \leq_M m_2$ and $\neg(a I m_1)$ together imply $\neg(a I m_2)$. For this reason, B is an order filter. The dual argument shows that A must be an order ideal. Since \equiv is a subrelation of I we clearly have $B = A' \subseteq A^\neq$. Suppose $m \in A^\neq$, $m \notin B$. Then there is some $a \in A$ such that $a I m$, and some $g \in G$ with $g \equiv m$. But $a I m \equiv g$ implies $g \leq_G A$, which implies $g \in A$. But that contradicts $m \in A^\neq$. \square

Theorem 1 *If (G, M, I) is a quasi-ordered context over \equiv , then*

$$(G \times M) \setminus J$$

is a closed subrelation of $(G, M, (G \times M) \setminus I)$ if and only if $I \subseteq J$ and (G, M, J) is quasi-ordered over \equiv .

Proof If J is a quasi-ordered super-relation of I , then Proposition 5 can be used to describe the concepts of $(G, M, (G \times M) \setminus J)$. It follows immediately from the definitions that the quasi-orders \leq_G and \leq_M induced by J contain the corresponding quasi-orders induced by I . Therefore order filters induced by J are also order filters for I , and the same holds for order ideals. As a consequence we get that each concept of $(G, M, (G \times M) \setminus J)$ also is a concept of $(G, M, (G \times M) \setminus I)$. This makes $(G \times M) \setminus J$ a closed relation.

For the converse assume that $\bar{J} := (G \times M) \setminus J$ is a closed subrelation of $(G, M, (G \times M) \setminus I)$. Then \bar{J} is the union of sets $A \times B$, where (A, B) is a concept of $(G, M, (G \times M) \setminus I)$. So if

$$g_1 \bar{J} m_2,$$

then there is some formal concept (A, B) of $(G, M, (G \times M) \setminus I)$ such that $g_1 \in A$ and $m_2 \in B$. Now consider arbitrary $g_2 \in G, m_1 \in M$ with $g_2 \equiv m_1$. We have

$$g_2 \in A \quad \text{or} \quad m_1 \in B,$$

because if $g_2 \notin A$ then we find some $b \in B$ with $g_2 \equiv b$ (because $A = B^\neq$), and $m_1 \equiv g_2 \equiv b$ enforces that $m_1 \in B$. So we have

$$g_2 \bar{J} m_2 \quad \text{or} \quad g_1 \bar{J} m_1.$$

We have proved

$$g_1 \bar{J} m_2 \Rightarrow \forall_{g_2 \equiv m_1} (g_1 \bar{J} m_1 \text{ or } g_2 \bar{J} m_2).$$

This is logically equivalent to

$$(\exists_{g_2 \equiv m_1} g_1 J m_1 \text{ and } g_2 J m_2) \Rightarrow g_1 J m_2,$$

or, in other notation,

$$g_1 J m_1 \equiv g_2 J m_2 \Rightarrow g_1 J m_2,$$

which is precisely the condition of being quasi-ordered for J . □

The theorem can be used to characterize the closed subrelations in the case of doubly founded completely distributive lattices. Such lattices are concept lattices of **contra-ordinal scales**, i.e., formal contexts of the form $(P, P, \not\leq)$, where (P, \leq) is some (quasi-)ordered set.

Corollary 1 *The closed subrelations of the contra-ordinal scale $(P, P, \not\leq)$, where (P, \leq) is some (quasi-)ordered set, are precisely of the form \sqsubseteq for quasi-orders \sqsubseteq containing \leq .*

Without proof we mention

Corollary 2 *A formal context, the extents of which are precisely the complete sublattices of $\mathfrak{B}(P, P, \not\leq)$, is*

$$(\mathfrak{B}(P, P, \not\leq), P \times P, \circ),$$

where

$$(A, B) \circ (p, q) \quad : \iff \quad \neg(p \in B \text{ and } q \in A).$$

The intents of this context are the quasi-orders containing \leq . An example is given in the Appendix.

4.2 Closed Subrelations of $\not\leq$

With the third corollary we come back to our original theme. Recall that the context $(G, M, \not\leq)$ is quasi-ordered over $\not\leq$.

Corollary 3 *Let (G, M, I) be a doubly founded reduced formal context. The closed subrelations of $(G, M, \not\leq)$ are precisely of the form $(G \times M) \setminus J$, where $J \subseteq G \times M$ is some relation containing $\not\leq$, for which (G, M, J) is quasi-ordered over $\not\leq$.*

The condition of being quasi-ordered is a closure condition. For given relations $R \subseteq G \times M$ and \equiv (as in 4.1) there is always a smallest relation S containing

R for which (G, M, S) is quasi-ordered over \equiv . In the case of the arrow relations, we write

$$\Downarrow = \text{trans}(\swarrow, \nearrow),$$

and, more generally, if D and U are relations containing \swarrow and \nearrow , respectively, then

$$(g, m) \in \text{trans}(D, U) : \iff \begin{cases} \text{there are } g_1 = g, g_2, \dots, g_k \in G \text{ and} \\ m_1, \dots, m_k, m_k = m \in M \text{ such that} \\ g_i D m_i \text{ holds for all } i \in \{1, \dots, k\} \text{ and} \\ g_i U m_{i-1} \text{ holds for all } i \in \{2, \dots, k\}. \end{cases}$$

Recall that each formal concept (A, B) of (G, M, \Downarrow) defines a compatible subcontext $(H, N, I \cap H \times N)$ of (G, M, I) by $H := G \setminus A$, $N := B$. If (A, B) is also a concept of the closed subrelation $(G \times M) \setminus \text{trans}(D, U)$, we will say that $(H, N, I \cap H \times N)$ is compatible with (D, U) .

Proposition 6 *A subcontext $(H, N, I \cap H \times N)$ is compatible with (D, U) if and only if*

- $h \in H$ and $h U m$ together imply $m \in N$, and
- $n \in N$ and $g D n$ together imply $g \in H$.

Proof Let us abbreviate

$$\overline{T} := (G \times M) \setminus \text{trans}(D, U).$$

“ \Rightarrow ”: If $(H, N, I \cap H \times N)$ is compatible with (D, U) , then $(G \setminus H, N)$ is a formal concept of (G, M, \overline{T}) . Now if $n \in N$ and $g D n$, then $(g, n) \in \text{trans}(D, U)$ and therefore $(g, n) \notin \overline{T}$. This implies $g \notin N^{\overline{T}} = G \setminus H$. Thus g must be in H .

If $h \in H$ and $h U m$, then choose some n and g such that $h \nearrow n$ and $g \swarrow m$. Such elements exist since (G, M, I) is reduced. Each subcontext compatible with (D, U) must in particular be compatible and therefore arrow-closed, thus $n \in N$ can be inferred from $h \nearrow n$. Moreover

$$g \swarrow m U h \nearrow n$$

implies $(g, n) \in \text{trans}(D, U)$ and thus $(g, n) \notin \overline{T}$. Then $g \notin N^{\overline{T}} = G \setminus H$, and consequently $g \in H$ and thus $m \in N$.

“ \Leftarrow ”: Conversely let $(H, N, I \cap H \times N)$ be a subcontext that satisfies the conditions of the proposition. We will show that $(G \setminus H, N)$ is a formal concept of (G, M, \overline{T}) . Let $n \in N$ and $(g, n) \notin \overline{T}$, thus $(g, n) \in \text{trans}(D, U)$. By the definition of $\text{trans}(D, U)$ there must be a sequence

$$g = g_1 D m_1 U g_2 D m_2 \dots U g_k D m_k = n.$$

Using the conditions along this sequence from right to left, and assuming $n \in N$, we get $g \in H$. Thus $(g, n) \in \overline{T}$ for all $g \notin H$, which proves that $G \setminus H \subseteq N^{\overline{T}}$. If $h \in H$ then there is some $m \in M$ such that $h \nearrow m$, which implies $m \in N$ and $(h, m) \notin \overline{T}$. Thus $G \setminus H = N^{\overline{T}}$.

It remains to show that $(G \setminus H)^{\overline{T}} \subseteq N$. So let $m \notin N$, and consider some $g \swarrow m$. $g \in H$ would contradict the first condition. Therefore $g \in G \setminus H$, $(g, m) \notin \overline{T}$ and thus $m \notin (G \setminus H)^{\overline{T}}$. \square

To find the closed relation characterizing concept algebra congruences, we may look for “additional arrow relations” \searrow and \swarrow , such that

$$\bar{J} = \text{trans}(\searrow \cup \swarrow, \nearrow \cup \nwarrow).$$

It turns out that we can actually split our considerations and treat the two unary operations separately. We will consider only one, say Δ .

5 Δ -Compatible Subcontexts

Definition 2 A compatible subcontext $(H, N, I \cap H \times N)$ of (G, M, I) is called Δ -compatible, if

$$(\varphi_{H,N}(A, B))^{\Delta} := \varphi_{H,N}((A, B)^{\Delta})$$

defines a unary operation on $\underline{\mathfrak{B}}(H, N, I \cap H \times N)$. ◇

From this condition it follows automatically that the so defined operation satisfies the equations which are satisfied by Δ .

Lemma 1 A compatible subcontext $(H, N, I \cap H \times N)$ of (G, M, I) is Δ -compatible iff

$$\forall n \in N \quad G \setminus n' \subseteq ((G \setminus n')'' \cap H)'.$$

Proof We must show that for arbitrary concepts (A_1, B_1) and (A_2, B_2) we have:

$$\text{if } \varphi(A_1, B_1) = \varphi(A_2, B_2) \text{ then } \varphi((A_1, B_1)^{\Delta}) = \varphi((A_2, B_2)^{\Delta}).$$

This can be simplified to the case that $A_2 = (A_1 \cap H)''$. The simplified condition then is that

$$\varphi((A, A')^{\Delta}) = \varphi(((A \cap H)'', (A \cap H)')^{\Delta})$$

holds for all concept extents A of (G, M, I) , which is equivalent to

$$(G \setminus A)' \cap N = (G \setminus (A \cap H)'')' \cap N \quad \text{for all extents } A.$$

Yet another equivalent reformulation is that for all extents A we have

$$\forall n \in N \quad (G \setminus A \subseteq n' \iff G \setminus (A \cap H)'' \subseteq n').$$

Since $(A \cap H)'' \subseteq A$, the direction \Leftarrow is always true and it suffices to prove

$$\forall n \in N \quad (G \setminus A \subseteq n' \Rightarrow G \setminus (A \cap H)'' \subseteq n'),$$

which is equivalent to

$$\forall n \in N \quad (G \setminus n' \subseteq A \Rightarrow G \setminus n' \subseteq (A \cap H)'').$$

For the special case that $A = (G \setminus n')''$ this is exactly the condition of the lemma. It remains to show that that condition is also sufficient.

Suppose therefore that A is some extent for which $G \setminus n' \subseteq A$. Since A is an extent, we infer $(G \setminus n')'' \subseteq A$, and consequently

$$(G \setminus n')'' \cap H \subseteq A \cap H.$$

If the condition of the lemma holds, then

$$G \setminus n' \subseteq ((G \setminus n')'' \cap H)'' \subseteq (A \cap H)'',$$

as was to be proved. \square

In order to better understand the condition of the lemma, let us abbreviate $E := G \setminus n'$. Then the condition is

$$E \subseteq (E'' \cap H)'',$$

which is clearly equivalent to

$$E'' = (E'' \cap H)'.$$

The latter condition states that the extent E'' has a generating system in H . A reformulation of the lemma therefore is:

Lemma 2 *A compatible subcontext $(H, N, I \cap H \times N)$ of (G, M, I) is \triangle -compatible iff for each attribute $n \in N$ the set H contains a generating set for the closure of $G \setminus n'$.*

It now becomes clearer how to define the “extra arrows”-relation λ : For each attribute $n \in N$ we must define λ in such a way that the transitive closure

$$\text{trans}(\swarrow \cup \lambda, \nearrow)$$

points from n to some generating set of $(G \setminus n')''$.

But there may be many generating sets, and it is not clear which one to choose in general. However, there is a class of lattices where the generating systems are essentially unique.

An **extremal point** of a concept extent A in a clarified context is an irreducible object $e \in A$ such that

$$e \notin ((A \cap G_{\text{irr}}) \setminus \{e\})''.$$

An extremal point of a subset is an extremal point of the extent it generates. Certainly, an extremal point of A must be contained in *every* generating set of A that consists of irreducibles. For certain lattices, the extremal points always form a generating set:

Theorem 2 ([1], Thm. 44) *In a finite meet-distributive lattice each concept extent is generated by its extremal points.*

Proposition 7 *If $\mathfrak{B}(G, M, I)$ is finite and meet-distributive, then a compatible subcontext $(H, N, I \cap H \times N)$ with $H \subseteq G_{\text{irr}}$, $N \subseteq M_{\text{irr}}$ is also \triangle -compatible iff H contains for each $n \in N$ all extremal points of $G \setminus n'$.*

Proof This is immediate from Theorem 2 together with Lemma 2. \square

In the meet-distributive case it is now clear how to define the relation \succ :

$$g \succ m : \Longleftrightarrow g \text{ is an extremal point of } G \setminus m'.$$

With this notation we get from Propositions 6 and 7

Theorem 3 *If $\mathfrak{B}(G, M, I)$ is finite and meet-distributive, then the congruence lattice of the concept algebra $\mathfrak{A}(G, M, I)$ is isomorphic to the concept lattice*

$$(G_{irr}, M_{irr}, G_{irr} \times M_{irr} \setminus \text{trans}(\swarrow \cup \succ, \nearrow)).$$

Combining this with the dual relation

$$g \prec m : \Longleftrightarrow m \text{ is extremal in } M \setminus g'$$

we get

Theorem 4 *The complete congruences of a finite distributive concept algebra $\mathfrak{A}(G, M, I)$ are given by those compatible subcontexts of*

$$(G_{irr}, M_{irr}, I \cap G_{irr} \times M_{irr}),$$

which are both \succ -closed and \prec -closed. The congruence lattice is isomorphic to the concept lattice of

$$(G_{irr}, M_{irr}, G_{irr} \times M_{irr} \setminus \text{trans}(\swarrow \cup \succ, \nearrow \cup \prec)).$$

We close our considerations by determining the congruence lattice of the concept algebra in Figure 1. Figure 3 shows four contexts. The first is the reduced context for Figure 1, with the arrow relations. The second displays the relations \succ and \prec . The third context is $(G, M, \text{trans}(\swarrow \cup \succ, \nearrow \cup \prec))$. The fourth finally is the complementary context of the third, which by Theorem 4 describes the concept algebra congruences.

	a	b	c	d
1	\swarrow	x	x	x
2	x	\swarrow	x	x
3	x	x	\swarrow	x
4	x	x	x	\swarrow

	a	b	c	d
1				
2				
3			\times	\times
4			\times	\times

	a	b	c	d
1	\times			
2		\times		
3			\times	\times
4			\times	\times

	a	b	c	d
1		\times	\times	\times
2	\times		\times	\times
3	\times	\times		
4	\times	\times		

Fig. 3. Determining the congruence lattice of the concept algebra in Figure 1.

The congruence lattice obviously is an eight element Boolean lattice. Consider, as an example, the formal concept $(\{1\}, \{b, c, d\})$ of the fourth context. It corresponds to the compatible subcontext $(\{2, 3, 4\}, \{b, c, d\})$. The induced congruence has the classes

$$\{0, 1\}, \{2, 5\}, \{3, 6\}, \{4, 7\}, \{8, 11\}, \{9, 12\}, \{10, 13\}, \{14, 15\}.$$

It can easily be read off from Figure 2 that this congruence indeed is compatible with the dicomplementation.

Appendix: All Complete Sublattices of a Small Lattice

As an example to Corollary 2, consider the lattice in Figure 4. Its 35 complete sublattices are the extents of the concept lattice in Figure 5.

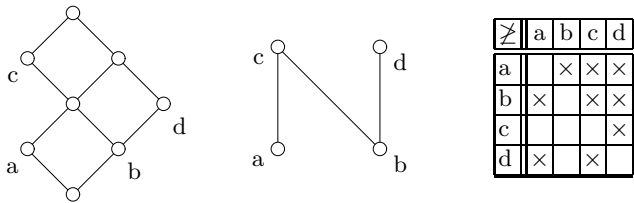


Fig. 4. A small distributive lattice, its ordered set (P, \leq) of irreducibles, and its standard context $(P, P, \not\leq)$.

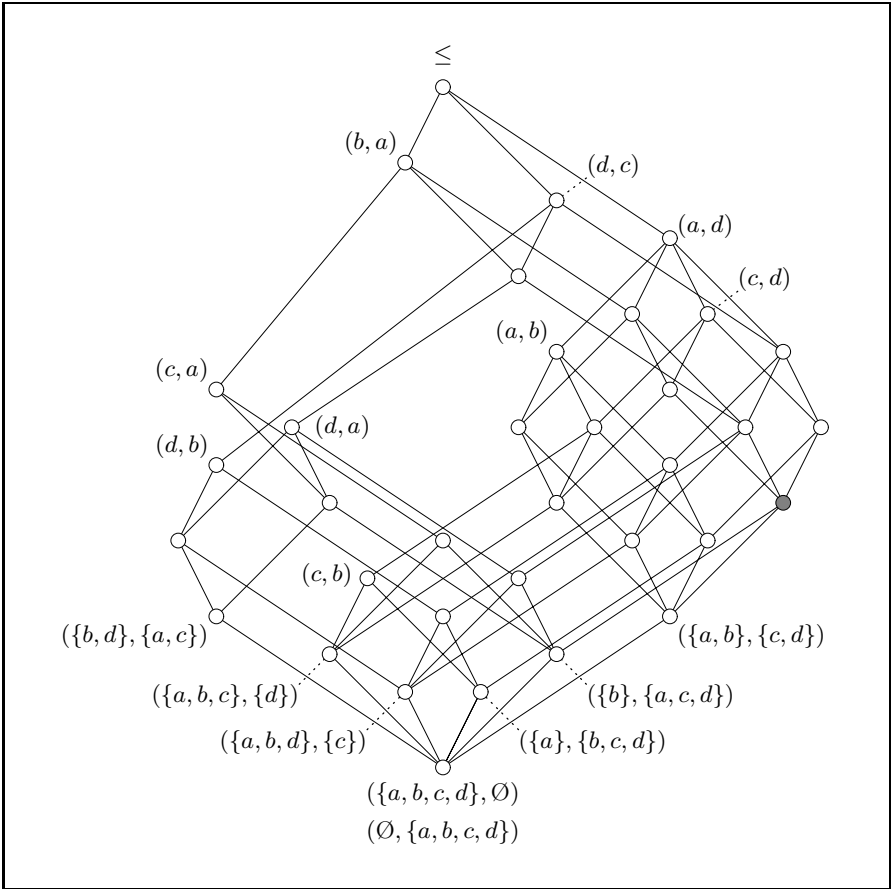


Fig. 5. The lattice of complete sublattices of the lattice in Figure 4. The shaded element serves as an example of how to interpret this lattice. Its meaning is explained in Figure 6.

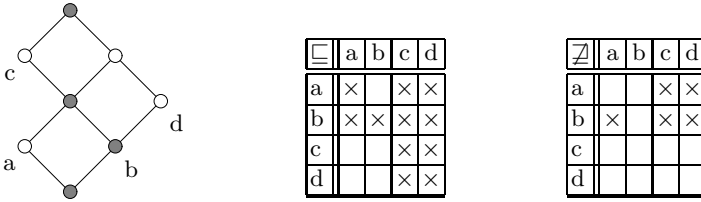


Fig. 6. With reference to the shaded element in Figure 5: its extent, its intent, and the corresponding closed subrelation of $(P, P, \not\leq)$.

References

1. B. Ganter & R. Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer Verlag 1999.
2. Léonard Kwuida, *Ph.D. thesis on dicomplemented lattices*. Dresden, 2004.
3. Bernhard Ganter, Léonard Kwuida, *Representing weak dicomplementations on finite distributive lattices*. Preprint, TU Dresden, 2002.
4. Rudolf Wille, *Boolean Concept Logic*. In: B. Ganter and G.W. Mineau (Eds.), *Conceptual Structures: Logical, Linguistic, and Computational Issues*. Springer LNAI 1867 (2000).

When Is a Concept Algebra Boolean?

Léonard Kwuida*

Institut für Algebra, TU Dresden
D-01062 Dresden

Abstract. Concept algebras are concept lattices enriched by a weak negation and a weak opposition. The introduction of these two operations was motivated by the search of a negation on formal concepts. These weak operations form a weak dicomplementation. A weakly dicomplemented lattice is a bounded lattice equipped with a weak dicomplementation. (Weakly) dicomplemented lattices abstract (at least for finite distributive lattices) concept algebras. Distributive double p-algebras and Boolean algebras are some special subclasses of the class of weakly dicomplemented lattices. We investigate in the present work the connection between weak dicomplementations and complementation notions like semicomplementation, pseudocomplementation, complementation or orthocomplementation.

1 Introduction

As a part of his project to extend *Formal Concept Analysis* to a broader field (called “*Contextual Logic*”), Rudolf Wille suggested and started a systematic study of potential “conceptual negations”. One of the starting point is that of Boolean algebras, which are most important and useful in *Propositional Calculus*. Is there a natural generalization of Boolean algebras to concept lattices? Yes, there is, and it has been introduced by Wille under the name of “*Concept Algebras*”.

At present, Concept Algebras are studied as mathematical objects. It turned out that even their elementary mathematical description raises a lot of problem, some of which are still unsolved. Here we give some solutions. They pave the ground for a clear algebraic theory, which is a precondition for (later) practical applications. For the moment, we know no practical examples where Concept Algebras are absolutely needed. However they generalize Boolean algebras in such a natural way, that it seems promising to develop their mathematical theory.

The aim of *Formal Concept Analysis* is among others to support human thinking. It is based on concept as unit of thought. Therefore we need a convenient logic (Concept Logic) in order to do reasoning on concepts. In [Bo54] George Boole developped a mathematical theory for logic. The encoding of the conjunction, the disjunction, the negation, the universe and “nothing” leads to the so called Boolean algebras. In the case of concepts the conjunction and disjunction are respectively encoded by the infimum and supremum operations of

* Research supported by DAAD.

the concept lattice. The universe is encoded by the greatest element of the concept lattice while the least element encodes “nothing”. The main problem is the lack of negation at the conceptual level since the complement of an extent is not always an extent.

We consider for example the interordinal scale $\mathbb{I}_n := (n, n, \leq) || (n, n, \geq)$ for $n = 4$. The corresponding context is on Figure 1 below.

	≤ 1	≤ 2	≤ 3	≤ 4	≥ 1	≥ 2	≥ 3	≥ 4
1	x	x	x	x	x			
2		x	x	x	x	x		
3			x	x	x	x	x	
4				x	x	x	x	x

Fig. 1. Context of the interordinal scale \mathbb{I}_4

This context is object reduced. Its concept lattice is drawn on Figure 2 below.

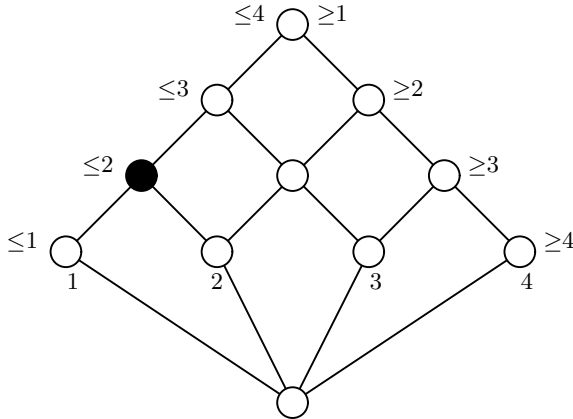


Fig. 2. Concept lattice of \mathbb{I}_4

Now let us examine the dotted concept of Figure 2. Its intent is $\{\leq 2\}$ and its extent $\{1, 2\}$. Since the attribute ≤ 2 implies ≤ 3 and ≤ 4 we decide to call this concept ≤ 2 . We want to negate this concept. If this concept has a negation, this should again be a concept of the same context. This should also be computed by a relatively simple process in this context. We would expect some laws such as “the law of double negation”, “the principle of excluded middle” or “the principle of contradiction” to follow from this computation process. How can we define such a process? How can we compute a *negation* of the concept ≤ 2 if there is one? One possibility is to consider the concept generated by the complement of the extent of the initial concept. Let us denote by Δ this com-

putation process. This gives for the concept ≤ 2 the concept ≥ 3 . i.e. $\leq 2^\Delta = \geq 3$. This makes sense as long as there is no object between 2 and 3. Now assume that there are some. For example we add a new object 2.5 and keep the set of attributes. The structure of the concept lattice does not change since the object 2.5 is the supremum of 2 and 3, and then reducible. But now the concept generated by the complement of the concept ≤ 2 is the concept ≥ 2 . The meet of these two concepts is not empty. We cannot call the concept ≥ 2 the negation of the concept ≤ 2 .

Note that the lattice on Figure 2 is dually pseudocomplemented¹. The dual pseudocomplementation sends the concepts ≤ 1 , ≤ 2 and ≤ 3 all together to ≥ 4 and sends the concepts ≥ 4 , ≥ 3 and ≥ 2 to ≤ 1 . It is less suitable if we wish to distinguish much more elements. If we require that the *negation* should be a complementation we would not be able to get the negation of concept that do not have a complement. In the present example only the concepts ≤ 1 , ≥ 4 , the smallest and the top element have a complement.

Note that the concept x^Δ is obtained from the concept x by a *type of opposing derivation*, meaning *what remains after removing* the concept x . This process is “in some sense not far from negation”. The binary relation $x^\Delta = y^\Delta$ defined on the concept lattice on Figure 2 is an equivalence relation. Though it is not a congruence relation, the operation $^\Delta$ on the quotient satisfied “the law of double negation”, “the principle of excluded middle” and “the principle of contradiction”.

Following Boole’s idea of a negation, and the requirement that the negation of a concept should of course be a concept, we consider the concept generated by the complement of the extent. i.e. In the case of a concept (A, B) we take the concept which extent is generated by \bar{A} ; that is the concept (\bar{A}'', \bar{A}') . Although the supremum $(A, B) \vee (\bar{A}'', \bar{A}')$ is 1, we may have $(A, B) \wedge (\bar{A}'', \bar{A}') \neq 0$. In every context, each concept is determined by its extent or its intent. Instead of the above approach with extents, we can also work with intents. Unfortunately the two operations obtained are in general different. A dream is for example to fall into the situation of Boolean algebras if the two operations are equal.

2 Weak Dicomplementation

2.1 Definition and Motivation

Definition 1. A **weakly dicomplemented lattice** is a bounded lattice L equipped with two unary operations $^\Delta$ and $^\nabla$ called **weak complementation** and **dual weak complementation**, and satisfying for all $x, y \in L$ the following equations:

- | | |
|---|---|
| (1) $x^{\Delta\Delta} \leq x$, | (1') $x^{\nabla\nabla} \geq x$, |
| (2) $x \leq y \implies x^\Delta \geq y^\Delta$, | (2') $x \leq y \implies x^\nabla \geq y^\nabla$, |
| (3) $(x \wedge y) \vee (x \wedge y^\Delta) = x$, | (3') $(x \vee y) \wedge (x \vee y^\nabla) = x$, |

¹ See Definition 5. The reader is also referred to [Gr70] for a better introduction to pseudocomplemented lattices.

The pair (x^Δ, x^∇) is called the **weak dicomplement** of x and the pair (Δ, ∇) a **weak dicomplementation**. We call x^Δ a **weak complement** of x and x^∇ a **dual weak complement** of x . The algebra $(L, \wedge, \vee, \Delta, 0, 1)$ is called a **weakly complemented lattice** while $(L, \wedge, \vee, \nabla, 0, 1)$ is called a **dual weakly complemented lattice**.

The motivation comes from concept algebras. The **concept algebra** of a formal context² \mathbb{K} is its concept lattice equipped with two unary operations Δ and ∇ called **weak negation** and **weak opposition**, and defined for each formal concept (A, B) by

$$(A, B)^\Delta := (\bar{A}'', \bar{A}') \quad \text{and} \quad (A, B)^\nabla := (\bar{B}', \bar{B}''),$$

where $\bar{A} := G \setminus A$ and $\bar{B} := M \setminus B$. These operations satisfy the equations in Definition 1 (see [Wi00]). Thus concept algebras are examples of weakly dicomplemented lattices. If a weakly dicomplemented lattice is a concept algebra of some context, it is said to be representable (by this context).

We would like to discover the equational or quasi-equational theory of concept algebras. An abstract structure (defined by a set of equations or implications) which satisfies all equations or quasi-equations valid in all concept algebras will be called a **dicomplemented lattice**. Since we are not sure that the equations in Definition 1 are enough to do the job, we prefer to use the term “*weakly dicomplemented lattice*”. At least for finite distributive lattices, there is no need to distinguish between both notions. This is proved by Ganter and the author in [GK02].

Here are some simple examples of (weakly) dicomplemented lattices. Obviously *Concept algebras* are dicomplemented lattices. As we expect *Boolean algebras* can be made into weakly dicomplemented lattices by defining $x^\Delta := x'$ and $x^\nabla := x'$, where x' is the complement of x in the Boolean algebra. Each *bounded lattice* L can be endowed with a **trivial dicomplementation** by defining $(1, 1)$, $(0, 0)$ and $(1, 0)$ as the (weak) dicomplement of 0, 1 and of each $x \notin \{0, 1\}$, respectively. The corresponding formal context is (L, L, \leq) .

Now we recall some unary operation properties we will often refer to.

Definition 2. Let $f : L \rightarrow L$ be a unary operation on a lattice L .

- (i) f is **monotone** if for all x and y the inequality $x \leq y$ implies $fx \leq fy$.
- (ii) f is **antitone** if for all x and y in L the inequality $x \leq y$ implies $fx \geq fy$.
- (iii) The operation f is called **square-extensive** if $x \leq f^2x$ for all $x \in L$, and **square-intensive** if $x \geq f^2x$ for all $x \in L$.
- (iv) An **involution** is a unary operation which is both square-extensive and square-intensive.
- (v) f is **idempotent** if $f^2x = fx$.

Definition 3. A **closure operator** on L is a monotone and idempotent mapping $f : L \rightarrow L$ such that $x \leq fx$. An element $x \in L$ is said to be *closed* (with

² For basic notions of Formal Concept Analysis the reader is referred to [GW99].

respect to f) if $x = fx$. Dually an **interior operator** on L is a monotone and idempotent mapping $g: L \rightarrow L$ such that $x \geq gx$. An element $x \in L$ is an interior element (with respect to g) if $x = gx$.

The weakly dicomplemented lattices form an equational class. Some basic properties are gathered in the following propositions:

Proposition 1 ([Wi00]). *Let L be a weakly dicomplemented lattice.*

1. *The mapping $\phi: x \mapsto x^{\Delta\Delta}$ is an interior operator on L . Dually the mapping $\psi: x \mapsto x^{\nabla\nabla}$ is a closure operator on L .*
2. *$x^{\nabla\nabla\nabla} = x^{\nabla} \leq x^{\Delta} = x^{\Delta\Delta\Delta}$.*
3. *$x^{\Delta\nabla} \leq x^{\Delta\Delta} \leq x \leq x^{\nabla\nabla} \leq x^{\nabla\Delta}$.*

The set $\bar{S}(L) := \{x \in L \mid x^{\Delta\Delta} = x\}$ of interior elements (also called **dual skeleton**) is equal to the set $\{x \in L \mid \exists_{y \in L} x = y^{\Delta}\}$ which is the same as the set $\{x \in L \mid x \wedge x^{\Delta} \leq x^{\Delta\Delta}\}$. Dually, the set $S(L) := \{x \in L \mid x^{\nabla\nabla} = x\}$ of closed elements (also called the **skeleton**) is equal to the set $\{x \in L \mid \exists_{y \in L} x = y^{\nabla}\}$ which is the same as the set $\{x \in L \mid x \vee x^{\nabla} \geq x^{\nabla\nabla}\}$.

Proposition 2. *For any weak dicomplementation $(^{\Delta}, ^{\nabla})$ on L , for all x and y in L the following properties hold:*

- | | |
|---|--|
| (4) $x \vee x^{\Delta} = 1$, | (4') $x \wedge x^{\nabla} = 0$, |
| (5) $(x \wedge x^{\Delta})^{\Delta} = 1$, | (5') $(x \vee x^{\nabla})^{\nabla} = 0$, |
| (6) $x^{\Delta} \leq y \iff y^{\Delta} \leq x$, | (6') $x^{\nabla} \geq y \iff y^{\nabla} \geq x$, |
| (7) $(x \wedge y)^{\Delta} = x^{\Delta} \vee y^{\Delta}$, | (7') $(x \vee y)^{\nabla} = x^{\nabla} \wedge y^{\nabla}$, |
| (8) $(x \wedge y)^{\Delta\Delta} \leq x^{\Delta\Delta} \wedge y^{\Delta\Delta}$, | (8') $(x \vee y)^{\nabla\nabla} \geq x^{\nabla\nabla} \vee y^{\nabla\nabla}$. |

Proof. (4) In Definition 1 setting x to 1 in the third equation gives $y \vee y^{\Delta} = 1$ for all $y \in L$.

(5) Since the operation $^{\Delta}$ is antitone we get $(x \wedge x^{\Delta})^{\Delta} \geq x^{\Delta} \vee x^{\Delta\Delta} = 1$.

(6) $x^{\Delta} \leq y \implies y^{\Delta} \leq x^{\Delta\Delta} \leq x$.

(7) Obviously $(x \wedge y)^{\Delta} \geq x^{\Delta} \vee y^{\Delta}$. If $a \geq x^{\Delta}$ and $a \geq y^{\Delta}$ then

$$a^{\Delta} \leq x^{\Delta\Delta} \wedge y^{\Delta\Delta} \leq x \wedge y. \text{ Thus } (x \wedge y)^{\Delta} \leq a^{\Delta\Delta} \leq a.$$

(8) This is trivial since $x \mapsto x^{\Delta\Delta}$ is monotone.

The remaining part follows dually.

The aim of this contribution is to clarify the connection between weak dicomplementation and some complementation notions. They are defined below.

2.2 Some Complementation Notions

We assume that L is a bounded lattice.

Definition 4. (i) A **semicomplementation** on L is a unary operation f such that for all $x \in L$, $x \wedge fx = 0$. The image of an element by a semicomplementation is called its *semicomplement*.

- (ii) A **dual semicomplementation** on L is a unary operation g such that for all x in L , $x \vee gx = 1$. The element gx said to be called a dual semicomplement of x .
- (iii) A **complementation** is a semi- and dual semicomplementation. The image of an element by a complementation is call its **complement**.
- (iv) L is said to be **uniquely complemented** if there is exactly one complementation on L . In this case we called it a **unique complementation**.
- (v) A **bi-uniquely complemented lattice** is a bounded lattice in which every element $x \notin \{0, 1\}$ has exactly two complements.

Each weak complementation is, by Proposition 2, a dual semicomplementation and each dual weak complementation a semicomplementation. The lattices on Figure 3 are bi-uniquely complemented. The antitone involution $x \mapsto x^\perp$ on B_4 is neither a weak complementation nor a dual weak complementation. In fact

$$(b \wedge a) \vee (b \wedge a^\perp) \neq b \text{ and } (a \vee b) \wedge (a \vee b^\perp) \neq a.$$

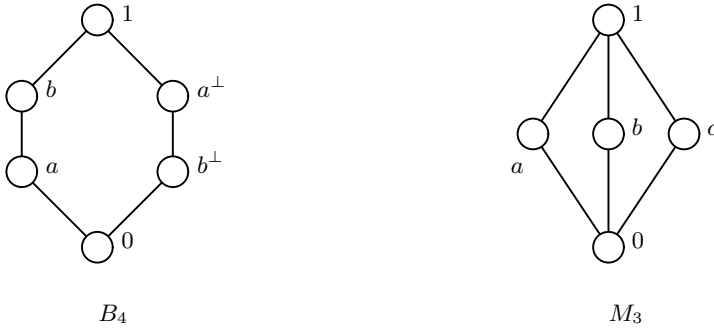


Fig. 3. Lattices with bi-unique complementation

In Figure 4 the element a has a bi-unique complement but these lattices are not bi-uniquely complemented. One might expect the pair (x^∇, x^Δ) in a weakly dicomplemented lattice to be a bi-unique complement of x . This cannot happen as we will see later.

On a weakly dicomplemented lattice ∇ is a **semicomplementation** while Δ is a **dual semicomplementation**. Note that both unary operations interchange 0 and 1. Moreover, x is comparable with x^Δ or x^∇ if and only if $\{x^\Delta, x^\nabla\}$ and $\{0, 1\}$ have a nonempty intersection.

Definition 5.

- (i) An element $x \in L$ has a **pseudocomplement**, denoted by x^* , if $x \wedge y = 0$ is equivalent to $y \leq x^*$. A **dual pseudocomplement** of $x \in L$ is an element x^+ (if it exists) such that $x \vee y = 1 \iff y \geq x^+$.
- (ii) A **pseudocomplemented lattice** (resp. **dual pseudocomplemented lattice**) is a lattice in which each element has a pseudocomplement (resp. a dual pseudocomplement). In these cases, $x \mapsto x^*$ (resp. $x \mapsto x^+$) is a unary operation called **pseudocomplementation** (resp. **dual pseudocomplementation**).

- (ii) A **p-algebra** is a lattice with a pseudocomplementation and a **dual p-algebra** a lattice with a dual pseudocomplementation.
- (iii) A **double p-algebra**³ is a pseudocomplemented and dually pseudocomplemented lattice. We will always denote pseudocomplementation and dual pseudocomplementation by $*$ and $^+$, respectively.

Each distributive double p-algebra is a weakly dicomplemented lattice. Not all double p-algebras are weakly dicomplemented lattices. The lattice on the left of Figure 4, with $a^{**} = a^{++} = a^{*+} = a^{+*} = a$, is a double p-algebra and is not a weakly dicomplemented lattice. However $(N_5; \wedge, \vee, \triangle, \nabla, 0, 1)$ with

$$a^{\nabla\nabla} = a^{\triangle\triangle} = a, a^{\nabla\triangle} = 1 \text{ and } a^{\triangle\nabla} = 0$$

on the right of Figure 4 is a (weakly) dicomplemented lattice. The corresponding formal context is its standard context⁴.



Fig. 4. Double p-algebra and weak dicomplementation on N_5

The other complementation notions considered in this contribution are

Definition 6. An **orthocomplementation** on a bounded lattice L is an involutorial antitone complementation. An **orthocomplemented lattice** or **ortholattice**, for short, is a bounded lattice equipped with an orthocomplementation. An **orthomodular lattice** is ortholattice satisfying for all elements a and b

$$a \leq b \implies a \vee (a^\perp \wedge b) = b \quad (\text{orthomodular law})$$

where $x \mapsto x^\perp$ denoted the orthocomplementation. A **weak orthocomplementation** is a square-extensive antitone semicomplementation. A **weakly orthocomplemented lattice** is a bounded lattice with a weak orthocomplementation. Dually is defined a **dual weak orthocomplementation**.

³ The reader is referred to [Ka73] or to [GV98].

⁴ The attributes are the meet-irreducible elements, the objects the join irreducible elements and the incidence relation is induced by the order relation of the lattice.

All dual weak complementations are weak orthocomplementation and all weak complementations are dual weak orthocomplementation. Could a weak complementation or dual weak complementation be an orthocomplementation?

The implication in the previous definition is obviously true with \perp taken to be a weak complementation. Thus all weakly complemented lattices satisfy the orthomodular law. They are ortholattices if and only if the weak complementation is also a complementation. Let L be a lattice and $a, b \in L$ with $a < b$. Let $c \in [a, b]$. An element $d \in [a, b]$ is called a **relative complement** of c in the interval $[a, b]$ if $c \vee d = b$ and $c \wedge d = a$. A lattice is called **relatively complemented** if for every $a, b, c \in L$ with $a < b$ and $c \in [a, b]$, c has at least one relative complement in $[a, b]$. Of course if L has 0 and 1 and is relatively complemented then L is complemented. If L has neither 0 nor 1 a relative complement (if there is one) is neither a semicomplement nor a dual semicomplement. Therefore this notion might be far away from the notions of weak complement and dual weak complement.

3 Weak Dicomplementation and Complementation

We shall consider a weak dicomplementation (\triangle, ∇) on a bounded lattice L .

Proposition 3.

- (i) If \triangle is a complementation then \triangle is a pseudocomplementation.
- (ii) If ∇ is a complementation then ∇ is a dual pseudocomplementation.

Proof. (ii) is a dual of (i). We are going to prove (i). If \triangle is a complementation then $x \wedge x^\triangle = 0$ for all $x \in L$. Moreover if $y \in L$ and $x \wedge y = 0$ then

$$y = (x \wedge y) \vee (x^\triangle \wedge y) = x^\triangle \wedge y$$

and so $y \leq x^\triangle$. This means that \triangle is a pseudocomplementation on L .

Corollary 1. *The operations \triangle and ∇ are unique complementations if and only if $\triangle = \nabla$.*

Obviously \triangle will be a complementation if it is a pseudocomplementation. This is also the case for ∇ if it is a dual pseudocomplementation. This seems to be an extreme situation. Contrary to what happens in those cases the operation \triangle seems to be closer to a dual pseudocomplementation than to a pseudocomplementation. Note that $x \wedge x^\triangle = 0$ and $x^\triangle = 1$ together imply $x = 0$.

Definition 7. *A weak complementation \triangle on L is said to be **nontrivial** if*

$$x^\triangle = 1 \implies x = 0 \text{ for all } x \in L.$$

A dual weak complementation ∇ is said to be nontrivial if

$$x^\nabla = 0 \implies x = 1 \text{ for all } x \in L.$$

A nontrivial weak dicomplementation is a weak dicomplementation (\triangle, ∇) such that \triangle and ∇ are nontrivial.

Theorem 1. *Let \triangle be a weak complementation on a lattice L . The following assertions are equivalent:*

- (i) \triangle is a nontrivial weak complementation,
- (ii) \triangle is a complementation on L ,
- (iii) $(L, \wedge, \vee, \triangle, 0, 1)$ is a p -algebra,
- (iv) all elements of L are interior elements,
- (v) the mapping $x \mapsto x^\triangle$ is bijective.

Each of these assertions implies the following:

- (vi) the de Morgan laws hold for the weak complementation,
- (vii) atoms and coatoms have unique complement.

Proof.

We start with the implication (i) \implies (ii). We assume that \triangle is a nontrivial weak complementation. From the equality $(x \wedge x^\triangle)^\triangle = 1$ we get $x \wedge x^\triangle = 0$ and \triangle is a complementation on L since $x \vee x^\triangle = 1$ always holds. The implication (ii) \implies (iii) is immediate from Proposition 3. To prove that (iii) implies (iv), we consider an arbitrary element x in L . We have

$$x = (x \wedge x^\triangle) \vee (x \wedge x^{\triangle\triangle}) = (x \wedge x^{\triangle\triangle}).$$

Therefore inequality $x \leq x^{\triangle\triangle}$ holds, implying the equality since $x^{\triangle\triangle} \leq x$ is always true by definition. Thus x is an interior element.

For (iv) \implies (v) we have, for all x and y in L the implications

$$x^\triangle = y^\triangle \implies x^{\triangle\triangle} = y^{\triangle\triangle} \implies x = y$$

and the mapping $x \mapsto x^\triangle$ is injective. This mapping is trivially surjective since each $x \in L$ is equal to y^\triangle for $y := x^\triangle$.

(v) \implies (i) is straightforward since the mapping $x \mapsto x^\triangle$ is bijective.

We assume that one of the statements (i) to (v) holds.

- (vi) We are going to prove that the de Morgan laws hold for the weak complementation. Proposition 2 give us $(x \wedge y)^\triangle = x^\triangle \vee y^\triangle$ and $(x \vee y)^\triangle \leq x^\triangle \wedge y^\triangle$. But there is an $z \in l$ such that $x^\triangle \wedge y^\triangle = z^\triangle$ because of the surjection property. Thus

$$z = z^{\triangle\triangle} = (x^\triangle \wedge y^\triangle)^\triangle = x^{\triangle\triangle} \vee y^{\triangle\triangle} = x \vee y$$

and $(x \vee y)^\triangle = x^\triangle \wedge y^\triangle$.

- (vii) Let a be an element of $L \setminus \{0, 1\}$. The element a^\triangle is the pseudocomplement of a . If x is a complement of a then $x \leq a^\triangle$. If a is a coatom and $x < a^\triangle$ we will have $x^\triangle > a^{\triangle\triangle} = a$ since $x \mapsto x^\triangle$ is injective, and a is interior. This would imply $x^\triangle = 1$ since a is a coatom. As \triangle is nontrivial we get $x = 0$ forcing a to be 1. This is a contradiction. Thus $x = a^\triangle$ and a has a unique complement. Coatom complements are atoms.

Theorem 2. *On a doubly founded⁵ lattice L , a weak dicomplementation (\triangle, ∇) is nontrivial if and only if $\triangle = \nabla$. In this case L is a Boolean algebra.*

Proof. The concept lattice of the context $\mathbb{K}(L) := (J(L), M(L), \leq)$ is isomorphic to L . A relation \preceq is defined on the set of all weak dicomplementations on L by:

$$(\triangle_1, \nabla_1) \preceq (\triangle_2, \nabla_2) : \Longleftrightarrow x^{\triangle_1} \leq x^{\triangle_2} \text{ and } x^{\nabla_1} \geq x^{\nabla_2} \text{ for all } x \text{ in } L.$$

This relation, called “finer than”⁶, is an order relation with a greatest element, namely the trivial dicomplementation. It also admits a smallest element which is the dicomplementation of the formal context $\mathbb{K}(L)$. We denote it by $(\triangle_{\mathbb{K}(L)}, \nabla_{\mathbb{K}(L)})$. If L is not distributive then there is a sublattice of L isomorphic to N_5 or to M_3 . Thus L contains one of the lattices on Figure 5 as sublattice.

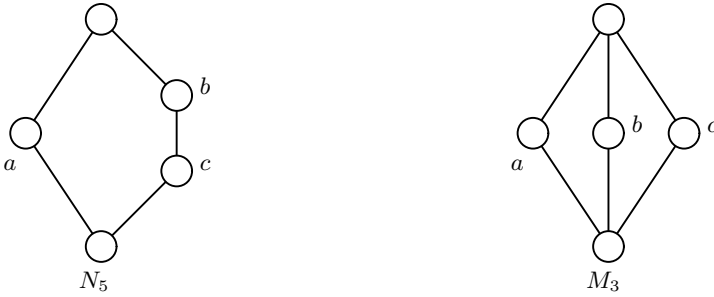


Fig. 5. Forbidden sublattices of distributive lattices

Observe that in the context $\mathbb{K}(L)$ we have for each element $x \in L$,

$$x^{\triangle_{\mathbb{K}(L)}} = (J(L) \setminus \downarrow x)'' = (\{u \in J(L) \mid u \not\leq x\})''.$$

Thus in both cases we will have for any weak complementation \triangle on L

$$c^\triangle \geq c^{\triangle_{\mathbb{K}(L)}} \geq a \vee b \geq c.$$

This would force c^\triangle to be equal to 1. This is impossible since our weak dicomplementation is nontrivial. Thus L is distributive. Consequently we get $\triangle = \nabla$ since they are both complementation on L . This achieves to prove that $(L, \wedge, \vee, \triangle, 0, 1)$ is a Boolean algebra.

We have also proved the following corollary.

⁵ The property of being doubly founded is a generalization of finiteness. We refer the reader to [GW99] for the definition of a doubly founded lattice. We need only the fact that the concept lattice of the context $(J(L), M(L), \leq)$ is isomorphic to L . Here $J(L)$ and $M(L)$ denote the set of join irreducible elements and meet irreducible elements respectively.

⁶ In [GK02] the authors proved that the set of all weak complementations on a doubly founded lattice endowed with the “finer than” relation is a complete lattice.

Corollary 2. *The operation \triangle is a nontrivial weak complementation on L if and only if $(L, \wedge, \vee, \triangle, 0, 1)$ is a Boolean algebra.*

Remark 1. If there is a negation on a context \mathbb{K} it should not depend on the intent side or extent side definition. Therefore the two operations should be equal.

$$(A, B)^\triangle = (A, B)^\nabla \iff (A, B)^\triangle \leq (A, B)^\nabla \iff \bar{A}'' \subseteq \bar{B}'$$

Thus any object g not in A has all attributes not in B . Each weakly dicomplemented lattice in which the two unary operations coincide is said to be **with negation**.

Another consequence of Theorem 1 is the following corollary.

Corollary 3.

- (i) $(L, \wedge, \vee, \triangle, \nabla, 0, 1)$ is a weakly dicomplemented lattice with negation if and only if $(L, \wedge, \vee, ', 0, 1)$ is a Boolean algebra, where $' := \triangle = \nabla$.
- (ii) $S(L)$ is a sublattice of L if and only if $(S(L), \wedge, \vee, \nabla, 0, 1)$ is a Boolean algebra. In particular $S(L) = L$ if and only if $(L, \wedge, \vee, \nabla, 0, 1)$ is a Boolean algebra.
- (ii') The dual of (ii) holds.

Proof.

- (i) It is enough to prove that under these conditions the operation \triangle is nontrivial. This is actually the case, since for each $x \in L$ we have

$$x^\triangle = 1 \implies x^\nabla = 1.$$

Thus $x \leq x^\nabla$ and $x = x \wedge x^\nabla = 0$.

- (ii) The dual weak complementation ∇ is nontrivial on $S(L)$.

We recall that each dual weakly complemented lattice is a weakly orthocomplemented lattice. It can be an ortholattice only if it is a Boolean algebra. A bi-uniquely complemented lattice cannot be distributive. Therefore the class of bi-uniquely complemented lattices intersects the class of weakly dicomplemented lattices only on the two element Boolean algebra. This class does not form a variety.

What happens if $(L, \wedge, \vee, \nabla, 0, 1)$ is a p-algebra? One might hope that L must be distributive. This is unfortunately not the case.

4 Dicomplementation and Double Pseudocomplementation

Definition 8. *The standard dicomplementation of a doubly founded lattice L is the dicomplementation induced by the concept algebra of its standard context.*

Theorem 3. *For doubly founded distributive lattices the double pseudocomplementation is the standard dicomplementation.*

Proof. Let L be a doubly founded distributive lattice. There is a poset (P, \leq) such that $(P, P, \not\leq)$ is isomorphic to the standard context of L . For a set of objects A (resp. attributes B), $A'' = \downarrow A$ (resp. $B'' = \uparrow B$) is the order ideal (resp. filter) generated by A (resp. B). Moreover we have

$$A' = \overline{\downarrow A} \text{ and } B' = \overline{\uparrow B}.$$

Let (A, B) be a concept; its weak opposition is

$$(A, B)^\nabla = (\bar{B}', \bar{B}'') = (\uparrow \bar{B}, \uparrow \bar{B}) = (\uparrow \bar{A}, \uparrow A).$$

The pseudocomplement of (A, B) , denoted by $(A, B)^*$, satisfies

$$\begin{aligned} (A, B)^* &= \bigvee \{(X, Y) \mid (A, B) \wedge (X, Y) = 0\} \\ &= \bigvee \{(X, Y) \mid A \cap X = \emptyset\} \\ &= \left(\left(\bigcup \{X \mid A \cap X = \emptyset\} \right)'' , \left(\bigcup \{X \mid A \cap X = \emptyset\} \right)' \right) \\ &= \left(\left(\bigcup \{\downarrow X \mid A \cap \downarrow X = \emptyset\} \right)'' , \left(\bigcup \{\downarrow X \mid A \cap \downarrow X = \emptyset\} \right)' \right) \\ &= \left(\bigcup \{\downarrow X \mid A \cap \downarrow X = \emptyset\}, \overline{\bigcup \{\downarrow X \mid A \cap \downarrow X = \emptyset\}} \right) \\ &= \left(\{x \mid A \cap \downarrow x = \emptyset\}, \overline{\{x \mid A \cap \downarrow x = \emptyset\}} \right) \end{aligned}$$

The set $\{x \mid A \cap \downarrow x = \emptyset\}$ is equal to $\uparrow \bar{A}$. This proves that $(A, B)^* \leq (A, B)^\nabla$. The reverse inequality is immediate since ∇ is a semicomplementation.

Dually $(A, B)^\Delta = (A, B)^+$, the dual pseudocomplement of (A, B) .

Remark 2. There are nondistributive finite double p -algebras which are dicomplemented lattices. An example for these is the lattice $L = 0 \oplus N_5 \oplus 1$ on the left of Figure 6. It is a double p -algebra. Its double pseudocomplementation is also the trivial dicomplementation although L is not distributive. However this dicomplementation is trivial. If a weak dicomplementation is assumed to be nontrivial then the distributivity follows from Theorem 1. What happens if the operation is neither trivial nor nontrivial?

Remark 3. The lattice M_4 with the antitone involution $^\perp$ is an orthomodular lattice. But the operation $^\perp$ is neither a weak complementation nor a dual weak complementation. On Figure 3 the lattice B_4 with the operation $^\perp$ is an ortholattice but is not orthomodular. The only (weak) dicomplementation on M_n , $n \geq 3$ is trivial. None of these lattices is pseudocomplemented.

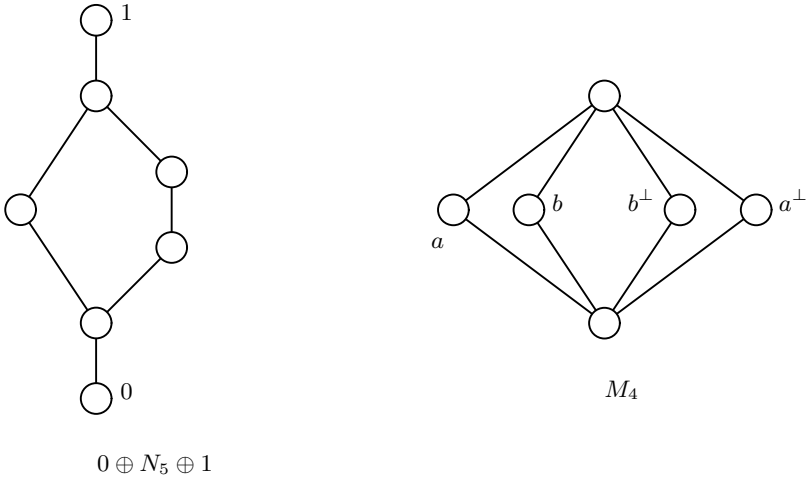


Fig. 6. Nondistributive dicomplemented lattice that is a double p-algebra. Orthomodular lattice

5 Conclusion and Further Research

We have explored the connection between weak dicomplementation and some extensions of the complementation of a Boolean algebra, arisen from a formalization of the “negation” by Boole. Not all extensions have been considered in this work. Boolean algebra extensions can be divided in three main groups. The first class retains only the distributivity. This is one of the most fruitfully investigated class of lattices⁷. Other extensions are more concerned with an abstraction of the negation. To these belong extensions by a single unary operation: p -algebras, Stone algebras, Ockham algebras, de Morgan algebras, Kleene algebras, weakly complemented lattices, orthocomplemented lattices and weakly orthocomplemented lattices. In the third class the negation is captured by two unary operations. Some members are double p-algebras, double MS -algebras and weakly dicomplemented lattices. An attribute exploration would better present the interdependence between these extensions. This is still to be done.

References

- [BD74] R. Balbes & P. Dwinger. *Distributive lattices*. University of Missouri Press. 1974.
- [Bo54] G. Boole. *An investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities*. Macmillan 1854. Reprinted by Dover Publ., New york (1958).

⁷ They are many monographs devoted to the theory of distributive lattices. [Gr70], [BD74], ...

- [GK02] B. Ganter & L. Kwuida. *Representing weak dicomplementations on finite distributive lattices*. Preprint MATH-AL-10-2002.
- [Gr70] G. Grätzer. *Distributive Lattices*. Springer 1970.
- [GV98] H. Gramaglia & D. Vaggione. *A note on distributive double p -algebras*. Czech. Math. J. 48, No.2, 321-327 (1998).
- [GW99] B. Ganter & R. Wille. *Formal Concept Analysis. Mathematical Foundations*. Springer 1999.
- [Ka73] T. Katriňák. *The structure of distributive double p -algebras. Regularity and congruences*. Algebra Universalis, Vol.3, fasc.2, 238-246 (1973).
- [Wi00] R. Wille. *Boolean Concept Logic* in B. Ganter & G.W. Mineau (Eds.) *Conceptual Structures: Logical, Linguistic, and Computational Issues*. Springer Proceedings 2000.

Background Knowledge in Concept Graphs

Frithjof Dau

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D-64289 Darmstadt
`dau@mathematik.tu-darmstadt.de`

Abstract. Traditional logic can be understood as the investigation of the three main essential functions of thinking – concepts, judgements and conclusions. In the last years, in a new research field termed *Contextual Logic*, a mathematical theory of this logic is elaborated. Concepts have already been mathematically elaborated by Formal Concept Analysis. Judgements and Conclusions can be expressed by so-called *Concept Graphs*, which are built upon families of formal contexts.

There are two approaches to concept graphs: A semantical approach, which investigates the theory of concept graphs in an algebraic manner, and a logical approach, which focuses on derivation rules for concept graphs, relying on a separation between syntax and semantics. In [24], Wille introduced two forms of complex implications (object implications and concept implications) to the semantical approach. In this paper it is investigated how these implications can be incorporated into the logical approach.

1 Introduction and Basic Definitions

The traditional philosophical goal of logic is the investigation of the forms of thinking. I. Kant explained this understanding of logic as “the theory of the three main essential functions of thinking – concepts, judgements and conclusions”. In the last years, a new research field termed *Contextual Logic (CL)* came up, where a mathematical theory of traditional logic is elaborated. The understanding of concepts in the line of traditional logic has already been adopted and successfully mathematized by Wille’s *Formal Concept Analysis (FCA)*, so the question of how to proceed with judgements and conclusions arose.

John Sowa developed on the basis of Peirce’s *Existential Graphs* and the semantic networks of artificial intelligence the theory of *Conceptual Graphs (CGs)*. These graphs are a diagrammatic system of logic whose purpose is ‘to express meaning in a form that is logically precise, humanly readable, and computationally tractable’ (see [16]). Thus, conceptual graphs can be considered to be formal representations of judgements. Moreover, Sowa incorporated deduction rules into his system which allow to obtain new graphs from given ones. Hence, this theory offers a formalization of conclusions as well.

Both FCA and CGs have been used for knowledge representation and processing. They have a common philosophical background, based on Peirce’s pragmatism. Particularly, a main goal of both systems is the support of human, rational

communication. Thus, for a mathematization of judgements and conclusions, associating FCA and CGs turned out to be a promising approach.

The combination of FCA and CGs yields mathematically defined structures termed *Concept Graphs*. The system of CGs is a very comprehensive and complex system, not mathematically defined, and without sharp borders, thus it has been impossible to develop a single system of Concept Graphs covering all features of CGs. Instead of that, since Concept Graphs were introduced for the first time in [20], different forms of concept graphs have been elaborated.

Most important for this paper is to distinguish two different accesses for a development of concept graphs, namely *semantical approaches*, and those based on a separation of syntax and semantics, termed *syntactical approaches*. In order to discuss this distinction accordingly, we first need some definitions. We start with the underlying structures for (both approaches to) concept graphs.

Definition 1 (Relational Graphs with Cuts).

A structure $(V, E, \nu, \top, \text{Cut}, \text{area})$ is called a relational graph with cuts if

- V , E and Cut are pairwise disjoint, finite sets whose elements are called vertices, edges and cuts, respectively,
- $\nu : E \rightarrow \bigcup_{k \in \mathbb{N}} V^k$ is a mapping¹,
- $\top \notin V \cup E \cup \text{Cut}$ is a single element called the sheet of assertion, and
- $\text{area} : \text{Cut} \cup \{\top\} \rightarrow \mathfrak{P}(V \cup E \cup \text{Cut})$ is a mapping such that
 - a) $c_1 \neq c_2 \Rightarrow \text{area}(c_1) \cap \text{area}(c_2) = \emptyset$,
 - b) $V \cup E \cup \text{Cut} = \bigcup_{d \in \text{Cut} \cup \{\top\}} \text{area}(d)$,
 - c) $c \notin \text{area}^n(c)$ for each $c \in \text{Cut} \cup \{\top\}$ and $n \in \mathbb{N}$ (with $\text{area}^0(c) := \{c\}$ and $\text{area}^{n+1}(c) := \bigcup \{\text{area}(d) \mid d \in \text{area}^n(c)\}$).

For an edge $e \in E$ with $\nu(e) = (v_1, \dots, v_k)$ we set $|e| := k$ and $\nu(e)|_i := v_i$. Sometimes, we will write $e|_i$ instead of $\nu(e)|_i$, and $e = (v_1, \dots, v_k)$ instead of $\nu(e) = (v_1, \dots, v_k)$. We set $E^{(0)} := V$ and $E^{(k)} := \{e \in E \mid |e| = k\}$ for $k > 0$.

As for every $x \in V \cup E \cup \text{Cut}$ we have exactly one context $c \in \text{Cut} \cup \{\top\}$ with $x \in \text{area}(c)$, we can write $c = \text{area}^{-1}(x)$ for every $x \in \text{area}(c)$, or even more simple and suggestive: $c = \text{cut}(x)$.

If $\text{Cut} = \emptyset$, we will speak simply of relational graphs, which can be identified with the more simple structure $\mathfrak{G} := (V, E, \nu)$.

The contextual structures for both approaches to concept graphs are so-called *power context families (PCFs)*, i.e., families $\mathbb{K} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \dots, \mathbb{K}_n)$ of contexts $\mathbb{K}_k := (G_k, M_k, I_k)$ with $G_0 \neq \emptyset$ and $G_k \subseteq (G_0)^k$ for each $k = 1, \dots, n$. The formal concepts of \mathbb{K}_k with $k = 1, \dots, n$ are called *relation concepts* because their extents are k -ary relations on the object set G_0 .

Semantical theories deal with the elaboration of a mathematical structure theory of concept graphs of a given power context family in an *algebraic* manner. In this sense, a *semantical concept graph of a power context family* is a structure $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ such that

¹ We set $\mathbb{N} := \{1, 2, 3, \dots\}$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.

1. (V, E, ν) is a relational graph without cuts,
2. $\kappa : V \dot{\cup} E \rightarrow \bigcup_{k \in \mathbb{N}_0} \mathfrak{B}(\mathbb{K}_k)$ is a mapping with $\kappa(u) \in \mathfrak{B}(\mathbb{K}_k)$ for $u \in E^{(k)}$,
and
3. $\rho : V \rightarrow G_0$ is a mapping such that $\rho(v) \in \text{Ext}(\kappa(v))$ for each $v \in V$, and
 $(\rho(e_1), \dots, \rho(e_k)) \in \text{Ext}(\kappa(e))$ for each $e = (v_1, \dots, v_k) \in E$.

Semantical concept graphs are mathematical structures defined over a given PCF. Then the forms and relations of those concept graphs are investigated. This includes *operations* on those graphs and a thorough study of the properties of the corresponding algebra of concept graphs of a given power context family. This approach was proposed by Wille in [20], and since then, it was further elaborated by himself, Pollandt and Schoolmann (see for instance [10,11,14]).

In the *syntactical* approach to concept graphs, the graphs are investigated with methods known from *mathematical logic*. This approach has been investigated by Prediger, Klinger, and Dau (see for instance [12,13,6,7,2]). In this paper, we will focus on the approach of the author.

First of all, a separation between syntax and semantics is introduced. In order to do this, first the notion of an alphabet is provided: An *alphabet* is a triple $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ of disjoint sets $\mathcal{G}, \mathcal{C}, \mathcal{R}$ such that \mathcal{G} is a finite set whose elements are called *object names*, $(\mathcal{C}, \leq_{\mathcal{C}})$ is a finite ordered set with a greatest element \top whose elements are called *concept names*, and $(\mathcal{R}, \leq_{\mathcal{R}})$ is a family of finite ordered sets $(\mathcal{R}_k, \leq_{\mathcal{R}_k})$, $k = 1, \dots, n$ (for an $n \in \mathbb{N}$) whose elements are called *relation names*. We suppose that we have a special name $\doteq \in \mathcal{R}_2$ which is called *identity*. Finally, let $\leq_{\mathcal{G}}$ be the order on $\mathcal{G} \dot{\cup} \{*\}$ such that $*$ is the greatest element of $\mathcal{G} \dot{\cup} \{*\}$, and all elements of \mathcal{G} are incomparable.

Now we can obtain the query graphs from relational graphs by labelling the vertices and edges with names from our alphabet. For our purpose, we will consider particularly (*syntactical*) *concept graphs with cuts* (CGwCs) (see [2]): A structure $\mathfrak{G} := (V, E, \nu, \top, \text{Cut}, \text{area}, \kappa, \rho)$ is called *concept graph with cuts over the alphabet \mathcal{A}* , if

- $(V, E, \nu, \top, \text{Cut}, \text{area})$ is a relational graph with cuts²,
- $\kappa : V \cup E \rightarrow \mathcal{C} \cup \mathcal{R}$ is a mapping such that $\kappa(V) \subseteq \mathcal{C}$, $\kappa(E) \subseteq \mathcal{R}$, and all $e \in E$ with $|e| = k$ satisfy $\kappa(e) \in \mathcal{R}_k$, and
- $\rho : V \rightarrow \mathcal{G} \dot{\cup} \{*\}$ is a mapping.

In contrast to semantical concept graphs, the vertices and edges of a syntactical concept graph are now labelled with *names* for objects and concepts. Thus, we have no direct connection between graphs and PCFs. As usual in mathematical logic, the connection is established by interpreting the names of the alphabet in models. Therefore, in addition to PCFs, we need an interpretation. The resulting structures are the models for the logical approach to concept graphs. The are given as follows:

² An additional restriction is needed: $(V, E, \nu, \top, \text{Cut}, \text{area})$ must have so-called *dominating nodes*. Due to space limitations, a discussion of this is omitted here. The interested reader is referred to [2].

Definition 2 (Contextual Models).

Let $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ be an alphabet and \mathbb{K} be a power context family. Then we call the disjoint union $\lambda := \lambda_{\mathcal{G}} \dot{\cup} \lambda_{\mathcal{C}} \dot{\cup} \lambda_{\mathcal{R}}$ of the mappings $\lambda_{\mathcal{G}}: \mathcal{G} \rightarrow G_0$, $\lambda_{\mathcal{C}}: \mathcal{C} \rightarrow \mathfrak{B}(\mathbb{K}_0)$ and $\lambda_{\mathcal{R}}: \mathcal{R} \rightarrow \mathfrak{R}_{\mathbb{K}}$ a \mathbb{K} -interpretation of \mathcal{A} if $\lambda_{\mathcal{C}}$ and $\lambda_{\mathcal{R}}$ are order-preserving, and $\lambda_{\mathcal{C}}, \lambda_{\mathcal{R}}$ satisfy $\lambda_{\mathcal{C}}(\top) = \top$, $\lambda_{\mathcal{R}}(\mathcal{R}_k) \subseteq \mathfrak{B}(\mathbb{K}_k)$ for all $k = 1, \dots, n$, and finally, $(g_1, g_2) \in \text{Ext}(\lambda_{\mathcal{R}}(\doteq)) \Leftrightarrow g_1 = g_2$ for all $g_1, g_2 \in G_0$. The pair (\mathbb{K}, λ) is called contextual model over \mathcal{A} or contextual structure over \mathcal{A} ³.

The focus of a logical investigation of concept graphs are *derivation rules* instead of operations. Particularly for CGwCs, a sound and complete calculus is provided in [2], to which we will refer in the remaining paper.

These two approaches are two different viewpoints on the same thing, thus they are not competing, but complementary to each other.

2 Background Knowledge

We have already seen that the syntactical concept graphs are based on an alphabet of which the names are ordered. This idea is adopted from the theory of Conceptual Graphs, where this order is usually called *type hierarchy*. This order encodes pre-knowledge on the concepts and relations we consider, thus, the type-hierarchy is how background knowledge is incorporated into the system of Conceptual Graphs. The type-hierarchy yields specific restrictions for models: The interpretation functions λ have to respect the type-hierarchy. As a simple example, let us consider two concept names C_1, C_2 with $C_1 \leq C_2$. If these concept names are interpreted in a contextual structure (\mathbb{K}, λ) , we demand that $\lambda_{\mathcal{C}}(C_1) \leq \lambda_{\mathcal{C}}(C_2)$ is satisfied. That is, $\forall g \in G_0 : g \in \text{Ext}(\lambda_{\mathcal{C}}(C_1)) \Rightarrow g \in \text{Ext}(\lambda_{\mathcal{C}}(C_2))$. Thus, $C_1 \leq C_2$ can be understood as an implication $C_1 \rightarrow C_2$ as well.

In [23,24], Rudolf Wille – inspired by Brandom’s elaboration of ‘material implications’ in [1] – introduced more complex implications as background knowledge to semantical concept graphs. First of all, he allowed *conjunctions* of concepts in the premises and conclusions of the implications (particularly, this approach is a generalization of the type-hierarchy). Furthermore, he used this approach for implications between objects as well. He considered the following implications:

For two given sets of objects (tuples) $A, B \subseteq G_k$ in a PCF $\mathbb{K} := (\mathbb{K}_0, \dots, \mathbb{K}_n)$ of contexts $\mathbb{K}_k = (G_k, M_k, I_k)$, a *object implication* $A \rightarrow B$ is given if we have $A^{I_k} \subseteq B^{I_k}$. For two given sets of concepts $\mathfrak{C}, \mathfrak{D} \subseteq \mathfrak{B}(\mathbb{K}_k)$, a *concept implication* $\mathfrak{C} \rightarrow \mathfrak{D}$ is given if we have $\bigwedge \mathfrak{C} \leq \bigwedge \mathfrak{D}$. In the following, it will be investigated how these kinds of implications can be integrated into the logical approach of concept graphs.

In order to exemplify object- and concept-implications, we adopt the example of [23]. Consider the context of Australian cities and states and its concept lattice, both depicted in Fig. 1. The state names are abbreviated by ACT

³ The name ‘contextual structure’ is chosen according to the term ‘relational structure’, which is common in first order logic.

implication, the right concept graph is more specific than the left one (where TERRITORY stands for the concept $\mu(territory)$).



Fig. 2. An example for dependencies of CGs due to an object implication

Analogously, due to the second concept implication, we have the following dependency:

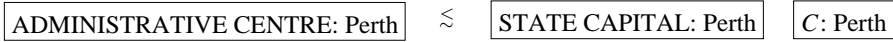


Fig. 3. An example for dependencies of CGs due to an concept implication

It should be briefly discussed why we focus on concept implications instead of attribute implications, as they are known in FCA. As said above, concept graphs are a main part of the program of Contextual Logic, of which the purpose is to develop and elaborate a formal theory of the elementary logic, based on the doctrines of concepts, judgements, and conclusions. Concept graphs are a formalization of judgements. The most elementary judgements are of the form “an object belongs to the extension of a concept”, and these judgements correspond to the “atomar” items of concept graphs, namely concept boxes and relation ovals (relation ovals represent elementary judgements where *tuples* of objects belong to the extension of a *relation*-concept). For this reason, in the framework of Contextual Logic, it is quite natural to elaborate concept implications instead of attribute implications. Nonetheless, as each concept is generated by attributes, i.e., as each concept is the meet of attribute concepts, concept implications can be in some sense reduced to attribute implications. We will come back to this point in Sec. 5, where the results of this paper are discussed.

As said above, Wille incorporated object- and concept-implications into his algebraic understanding of concept graphs. The goal of this paper is to incorporate these implications into the logical approach to concept graphs. In the remaining part of this section, some basic ideas for doing this shall be described.

Obviously, an (object or concept) implication $X \rightarrow Y$ holds if each implication $X \rightarrow y$, $y \in Y$ holds (the above examples are already of this type). Thus, it is sufficient to consider only implications with a simple conclusion. Next, due to the separation of syntax and semantics, we have to consider implication between *names* for objects and concepts. Similar to the type hierarchy from which we obtained specific restrictions of our models, these implications correspond to (further) restrictions of our models as well. The implications can be now restated as follows: If $g_1, \dots, g_n \in \mathcal{G}$ are object names, then $g_1 \wedge g_2 \wedge \dots \wedge g_{n-1} \rightarrow g_n$ is called a (*syntactical*) *object implication*. If $C_1, \dots, C_n \in \mathcal{C}$ are concept names, then $C_1 \wedge C_2 \wedge \dots \wedge C_{n-1} \rightarrow C_n$ is called a (*syntactical*) *concept implication*.

Note that we consider object implications only for objects, not for tuples, and concept implications only for concept names, not for relation names. The other cases can be handled analogously and are omitted for sake of simplicity.

If an object implication $g_1 \wedge g_2 \wedge \dots \wedge g_{n-1} \rightarrow g_n$ is given, we consider only models $\mathcal{M} := (\mathbb{K}, \lambda)$ which respect this implication, i.e., models which satisfy $\{\lambda_{\mathcal{G}}(g_1), \dots, \lambda_{\mathcal{G}}(g_{n-1})\}^{I_0} \subseteq \{\lambda_{\mathcal{G}}(g_n)\}^{I_0}$. In other words:

$$\mathcal{M} \text{ respects } g_1 \wedge \dots \wedge g_{n-1} \rightarrow g_n \quad \text{iff} \quad \gamma(\lambda_{\mathcal{G}}(g_1)) \vee \dots \vee \gamma(\lambda_{\mathcal{G}}(g_{n-1})) \geq \gamma(\lambda_{\mathcal{G}}(g_n)) \quad (1)$$

Analogously, if a concept implication $C_1 \wedge C_2 \wedge \dots \wedge C_{n-1} \rightarrow C_n$ is given, we have:

$$\mathcal{M} \text{ respects } C_1 \wedge \dots \wedge C_{n-1} \rightarrow C_n \quad \text{iff} \quad \lambda_{\mathcal{C}}(C_1) \wedge \dots \wedge \lambda_{\mathcal{C}}(C_{n-1}) \leq \lambda_{\mathcal{C}}(C_n) \quad (2)$$

If we understand a subsumption relation $C_1 \leq C_2$ in a type hierarchy as an implication $C_1 \rightarrow C_2$, Eqn. (2) extends the conditions for models which respect a type-hierarchy.

We have already seen that background knowledge yields specific restrictions for the models. These restrictions have to be reflected by derivation rules, i.e., we have to provide rules for the calculus which reflect exactly the background knowledge. For example, the calculus should allow us to derive the left graph of Fig. 3 from the right one. Analogously, the left graph of Fig. 2 should be derivable from the right one.

3 General Logical Background

As already mentioned, the graphs in [2] are based on an alphabet of which the names are ordered. Moreover, in [2], a sound and complete calculus for CGwCs is provided. Particularly, this calculus has two rules (named ‘specialization’ and ‘generalization’) which reflect the order of the names.

Object- and concept implications are more complex than an order on the set of object- or concept names, that is, an order on the set of names can be reflected by object- or concept implications, but not vice versa. Our goal is to incorporate these kinds of implications into the theory of CGwCs. To be more specific: Instead of an ordered alphabet of which the order is reflected by rules in the calculus, we will now have an unordered alphabet, but a set of implications which has to be reflected by the calculus. Of course, this calculus has not to be developed ‘from the scratch’: We already have a sound and complete calculus for CGwCs, but without background-knowledge in the form of object- or concept implications. These implications will be incorporated by adding further rules to the calculus. It has to be checked whether these additional rules correspond to the implications of our background knowledge. In this section, a method for doing this will be described.

Our starting point are CGwCs which are based on a (non-ordered) alphabet. These concept graphs are evaluated in contextual structures via the relation \models . Let us denote the class of models for this kind of concept graphs by \mathfrak{M}_1 . In [2],

a calculus for these graphs is provided. Let us denote the set of rules by \vdash_1 . (This use of the symbol ‘ \vdash ’ is a little bit sloppy: Usually, the symbol denotes the syntactical entailment relation between formulas of a given logic, which is *derived* from a set of rules. We will use ‘ \vdash ’ in this sense as well, but the set of rules shall also be denoted with ‘ \vdash ’. It will be clear from the context which use of ‘ \vdash ’ is intended.)

The basic idea is that further background knowledge corresponds to a restriction of \mathfrak{M}_1 (see the discussion above). So we get class of models $\mathfrak{M}_2 \subseteq \mathfrak{M}_1$. On the other hand, we are looking for an extension of the set of rules which captures exactly the background knowledge, i.e., we are looking for a calculus $\vdash_2 \supseteq \vdash_1$.

Similar to the distinction between \vdash_1 and \vdash_2 , we will distinguish between \models_1 and \models_2 as follows: For two graphs $\mathfrak{G}_a, \mathfrak{G}_b$, we will write $\mathfrak{G}_a \models_i \mathfrak{G}_b$ if and only if $\mathcal{M} \models \mathfrak{G}_b$ for all models $\mathcal{M} \in \mathfrak{M}_i$ with $\mathcal{M} \models \mathfrak{G}_a$.

Analogously, we define $\mathfrak{H} \vdash_i \mathfrak{G}$ and $\mathfrak{H} \models_i \mathfrak{G}$ for a graph \mathfrak{G} and a set of graphs \mathfrak{H} .

The soundness and completeness of \vdash_1 can be now stated as follows:

$$\text{Let } \mathfrak{G}_1, \mathfrak{G}_2 \text{ two CGwCs. We have: } \mathfrak{G}_1 \vdash_1 \mathfrak{G}_2 \iff \mathfrak{G}_1 \models_1 \mathfrak{G}_2 \quad (3)$$

We assume that both \vdash_1, \vdash_2 are Peirce-style-calculi, that is:

1. Each rule can be applied either in arbitrary positive contexts, or in arbitrary negative contexts, or in each context of a graph, and
2. For each rule which can be applied in arbitrary positive contexts, we have a rule in the opposite direction which can be applied in arbitrary negative contexts. Vice versa: For each rule which can be applied in arbitrary negative contexts, we have a rule in the opposite direction which can be applied in arbitrary positive contexts. The rules which can be applied in each contexts can be applied in both directions.

The calculus \vdash_1 , and hence \vdash_2 as well, encompasses the 5 basic-rules of Peirce. This yields a first simple lemma:

Lemma 1 (Syntactical and Semantical Deduction Theorem).

Let $\mathfrak{G}_a, \mathfrak{G}_b$ be two CGwCs. Then we have

$$\mathfrak{G}_a \vdash_i \mathfrak{G}_b \iff \vdash_i \left(\mathfrak{G}_a \quad \mathfrak{G}_b \right) \quad (4)$$

$$\mathfrak{G}_a \models_i \mathfrak{G}_b \iff \models_i \left(\mathfrak{G}_a \quad \mathfrak{G}_b \right) \quad (5)$$

Proof: We show both directions separately and start with ‘ \implies ’:

$$\begin{array}{c} \text{dc.} \\ \vdash \end{array} \left(\quad \quad \right) \xrightarrow{\text{ins.}} \left(\mathfrak{G}_a \quad \quad \right) \xrightarrow{\text{it.}} \left(\mathfrak{G}_a \quad \mathfrak{G}_a \right) \xrightarrow{\text{Lm. 6.4. in [2]}} \left(\mathfrak{G}_a \quad \mathfrak{G}_b \right)$$

The direction ' \Leftarrow ' is done as follows:

$$\mathfrak{G}_a \xrightarrow{\text{Lm. 6.4. in [2]}} \vdash \mathfrak{G}_a \left(\mathfrak{G}_a \quad \mathfrak{G}_b \right) \xrightarrow{\text{deit.}} \vdash \mathfrak{G}_a \left(\mathfrak{G}_b \right) \xrightarrow{\text{dc}} \vdash \mathfrak{G}_a \quad \mathfrak{G}_b \xrightarrow{\text{era.}} \vdash \mathfrak{G}_b$$

□

By definition of \vdash , we have: If $\mathfrak{H} := \{\mathfrak{G}_i \mid i \in I\}$ is a (possibly empty) set of CGwCs, then a graph \mathfrak{G} can be derived from \mathfrak{H} if and only if there is a finite subset $\{\mathfrak{G}_1, \dots, \mathfrak{G}_n\} \subseteq \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \vdash \mathfrak{G}$ (where $\mathfrak{G}_1 \dots \mathfrak{G}_n$ is the juxtaposition of $\mathfrak{G}_1, \dots, \mathfrak{G}_n$).

It is easy to check that \mathfrak{M}_1 satisfies the compactness-theorem. Thus, for the semantical entailment relation, we have a similar property, i.e. we have: If $\mathfrak{H} := \{\mathfrak{G}_i \mid i \in I\}$ is a (possibly empty) set of nonexistential concept graphs, then $\mathfrak{H} \models \mathfrak{G}$ if and only if there is a finite subset $\{\mathfrak{G}_1, \dots, \mathfrak{G}_n\} \subseteq \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \models \mathfrak{G}$. From this we immediately conclude for a CGwC \mathfrak{G} and set \mathfrak{H} of CGwCs:

$$\mathfrak{H} \vdash_1 \mathfrak{G} \iff \mathfrak{H} \models_1 \mathfrak{G} \quad (6)$$

Now the main idea for the extension of \vdash_1 to \vdash_2 is the following: The models in \mathfrak{M}_2 have to be described properly by the additional rules in $\vdash_2 \setminus \vdash_1$. That is, on the one hand, the rules in \vdash_2 have to be sound, i.e. for two CGwCs $\mathfrak{G}_1, \mathfrak{G}_2$ we have

$$\mathfrak{G}_1 \vdash_2 \mathfrak{G}_2 \implies \mathfrak{G}_1 \models_1 \mathfrak{G}_2 \quad (7)$$

On the other hand, let us assume that for each $\mathcal{M} \in \mathfrak{M}_1 \setminus \mathfrak{M}_2$, there exists a graph \mathfrak{G}_M with

$$\vdash_2 \mathfrak{G}_M \text{ and } \mathcal{M} \not\models \mathfrak{G}_M \quad (8)$$

If the last two assumptions (7) and (8) hold, we obtain that \vdash_2 is an adequate calculus, as the following theorem shows.

Theorem 1 (Completeness of \vdash_2).

\vdash_2 is complete, i.e. $\mathfrak{G}_a \models_2 \mathfrak{G}_b \implies \mathfrak{G}_a \vdash_2 \mathfrak{G}_b$.

Proof: Let $\mathfrak{H} := \{\mathfrak{G}_M \mid \mathcal{M} \in \mathfrak{M}_1 \setminus \mathfrak{M}_2\}$. From (7) we conclude: $\models_2 \mathfrak{G}_M$ for all $\mathfrak{G}_M \in \mathfrak{H}$. Now (8) yields:

$$\mathfrak{M}_2 = \{\mathcal{M} \in \mathfrak{M}_1 \mid \mathcal{M} \models \mathfrak{G} \text{ for all } \mathfrak{G} \in \mathfrak{H}\} \quad (9)$$

Thus, we get:

$$\begin{aligned} \mathfrak{G}_a \models_2 \mathfrak{G}_b &\stackrel{\text{Def}}{\iff} \text{f.a. } \mathcal{M} \in \mathfrak{M}_2 : \text{if } \mathcal{M} \models \mathfrak{G}_a, \text{ then } \mathcal{M} \models \mathfrak{G}_b \\ &\stackrel{(9)}{\iff} \text{f.a. } \mathcal{M} \in \mathfrak{M}_1 : \text{if } \mathcal{M} \models \mathfrak{G} \text{ for all } \mathfrak{G} \in \mathfrak{H} \text{ and } \mathcal{M} \models \mathfrak{G}_a, \\ &\quad \text{then } \mathcal{M} \models \mathfrak{G}_b \\ &\iff \mathfrak{H} \dot{\cup} \{\mathfrak{G}_a\} \models_1 \mathfrak{G}_b \\ &\stackrel{(6)}{\iff} \mathfrak{H} \dot{\cup} \{\mathfrak{G}_a\} \vdash_1 \mathfrak{G}_b \\ &\iff \text{there are } \mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H} \text{ with } \mathfrak{G}_1 \quad \mathfrak{G}_2 \dots \quad \mathfrak{G}_n \quad \mathfrak{G}_a \vdash_1 \mathfrak{G}_b \\ &\stackrel{(4)}{\iff} \text{there are } \mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H} \text{ with } \vdash_i \left(\mathfrak{G}_1 \mathfrak{G}_2 \dots \mathfrak{G}_n \mathfrak{G}_a \quad \mathfrak{G}_b \right) \end{aligned}$$

From this result, $\vdash_2 \supseteq \vdash_1$ and (8) we obtain:

$$\begin{array}{ccc} \vdash_2 \mathfrak{G}_1 \dots \mathfrak{G}_n \quad \left(\mathfrak{G}_1 \dots \mathfrak{G}_n \mathfrak{G}_a \left(\mathfrak{G}_b \right) \right) & \text{deit.} & \vdash_2 \mathfrak{G}_1 \\ & & \\ \dots \mathfrak{G}_n \quad \left(\mathfrak{G}_a \left(\mathfrak{G}_b \right) \right) & \text{era.} & \vdash_2 \left(\mathfrak{G}_a \left(\mathfrak{G}_b \right) \right) \end{array}$$

Now (4) yields $\mathfrak{G}_a \vdash_2 \mathfrak{G}_b$. □

4 Incorporating Background Knowledge into the Calculus

As the preceding section shows, we have to find additional and sound rules which describe properly the restriction of \mathfrak{M}_1 to \mathfrak{M}_2 . This is done separately for object- and concept-implications. Concept implications are easier to handle, so we start with them.

4.1 Incorporating Concept Implications

We have already seen that concept implications are a generalization of the order on the concept names in a type-hierarchy. A subsumption relation $C_1 \leq C_2$ is reflected in \vdash_1 by the rules ‘generalization’ and ‘specialization’, which –particularly for $C_1 \leq C_2$ – allow to replace an evenly enclosed vertex $\boxed{C_1 : g}$ (with $g \in \mathcal{G} \dot{\cup} \{*\}$) by $\boxed{C_2 : g}$, and, vice versa, to replace an oddly enclosed vertex $\boxed{C_2 : g}$ by $\boxed{C_1 : g}$. These rules will be canonically extended.

Let a concept implication $C_1 \wedge \dots \wedge C_{n-1} \rightarrow C_n$ for concept names $C_1, \dots, C_n \in \mathcal{C}$ be given. As discussed in Sec. 2, this implication yields the following restriction for models: $\mathcal{M} \in \mathfrak{M}_2 \iff \mathcal{M} \in \mathfrak{M}_1$ and $\lambda_{\mathcal{C}}(C_1) \wedge \dots \wedge \lambda_{\mathcal{C}}(C_{n-1}) \leq \lambda_{\mathcal{C}}(C_n)$. In other words: For all $g \in G_0$ we have

$$g \in \lambda_{\mathcal{C}}(C_1) \wedge \dots \wedge g \in \lambda_{\mathcal{C}}(C_{n-1}) \implies g \in \lambda_{\mathcal{C}}(C_n) \quad (10)$$

Before we provide a rule which encompasses this implication, we need a simple definition.

Definition 3 ($\theta_{\mathfrak{G}}$).

Let $\mathfrak{G} := (V, E, \nu, \top, \text{Cut}, \text{area}, \kappa, \rho)$ be a concept graph over \mathcal{A} . Let $\theta_{\mathfrak{G}}$ be the smallest equivalence relation on V such that if $e \in E$ is an edge with $\nu(e) = (v_1, v_2)$ and $\kappa(e) \leq \top$, then $v_1 \theta_{\mathfrak{G}} v_2$.

Now the new rules for the calculus are:

1. generalization of concepts

Let $\mathfrak{G} := (V, E, \nu, \top, \text{Cut}, \text{area}, \kappa, \rho)$ be a CGwCs and let $v_1, \dots, v_{n-1} \in V$ be vertices with $v_1 \theta_{\mathfrak{G}} v_2 \theta_{\mathfrak{G}} \dots \theta_{\mathfrak{G}} v_{n-1}$ such that $c := \text{cut}(v_1) = \dots = \text{cut}(v_{n-1})$ is an even context, and $\kappa(v_i) = C_i$ for all $1 \leq i \leq n-1$. Then for each v_i , $\kappa(v_i) = C_i$ may be replaced by $\kappa(v_i) = C_n$.

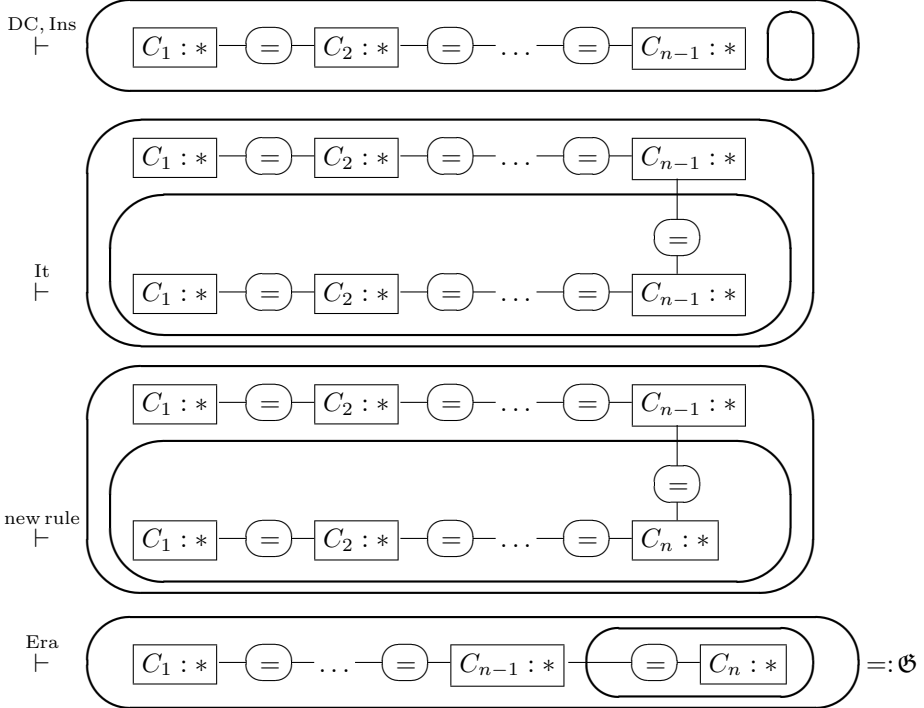
2. specialization of concepts

The preceding rule may be reversed in negative contexts.

We have to show that these rules satisfy Eqn. (7) and Eqn. (8).

The soundness of the rules can be proven analogously to the soundness of the rules ‘generalization’ and ‘specialization’ in \vdash_1 (see [2]), hence Eqn. (7) holds.

Now we derive with the new rule a graph \mathfrak{G} as follows:



Let $\mathcal{M} \in \mathfrak{M}_1 \setminus \mathfrak{M}_2$ be an arbitrary contextual model. Eqn. (10) immediately yields that $\mathcal{M} \not\models \mathfrak{G}$. We conclude that Eqn. (8) is satisfied, thus Thm. 1 yields that \vdash_2 is complete.

4.2 Background Knowledge for Objects

Incorporating concept implications into \vdash_2 was straight forward. First of all, concept implications can be considered to be a generalization of the idea of a type hierarchy, i.e., in our alphabet, we have an order on the concept names. But we do not have a corresponding order on the object names in a type hierarchy, which could analogously be generalized to object implications. Moreover, the proof for the completeness in the last subsection heavily relies on the fact that we can quantify over objects (to be more precisely: In the system of CGwCs, we can express universal quantification and material implications). Thus, the idea of the last subsection cannot be adopted for object implications, as we have no possibility to quantify over concepts.

Since quantification over concepts is not possible, we need ‘enough’ concept names to describe all concepts of a contextual structure. Then we can hopefully adopt the idea of the last section. First of all, we have to clarify the meaning of ‘enough’. Usually, the concept names do not encompass all concepts in a contextual structure. But it is not necessary that each concept (A, B) of a given contextual model is generated by a concept name (i.e., there exists a concept name C with $\lambda_C(C) = (A, B)$). The following further restriction for the models in \mathfrak{M}_2 will turn out to be sufficient:

General Restriction for Models $\mathcal{M} \in \mathfrak{M}_2$: The concepts of \mathcal{M} which are generated by concept names are \bigwedge -dense in the set of all concepts. (*)

Now let an object implication $g_1 \wedge \dots \wedge g_{n-1} \rightarrow g_n$ for object names $g_1, \dots, g_n \in \mathcal{G}$ be given. This implication yields the following restriction for models:

$$\mathcal{M} \in \mathfrak{M}_2 : \iff \mathcal{M} \in \mathfrak{M}_1, \mathcal{M} \text{ satisfies } (*) \text{ and } \lambda_{\mathcal{G}}(g_1) \vee \dots \vee \lambda_{\mathcal{G}}(g_{n-1}) \geq \lambda_{\mathcal{G}}(g_n)$$

The rules capturing this restriction are defined analogously to the rules for concept implications. They are as follows:

1. generalization of objects

Let $\mathfrak{G} := (V, E, \nu, \top, \text{Cut}, \text{area}, \kappa, \rho)$ be a CGwCs and let $v_1, \dots, v_n \in V$ be vertices such that $c := \text{cut}(v_1) = \dots = \text{cut}(v_{n-1}) = \text{cut}(v_n)$ is an even context, $\rho(v_i) = g_i$ for all $1 \leq i \leq n-1$, and there is a (fixed) $C \in \mathcal{C}$ with $\kappa(v_i) = C$ for all $1 \leq i \leq n-1$. Then for each v_i , $\rho(v_i) = g_i$ may be replaced by $\rho(v_i) = g_n$.

2. specialization of objects

The preceding rule may be reversed in negative contexts.

Again, the soundness of these rules can be proven analogously to the soundness of the rules ‘generalization’ and ‘specialization’ in \vdash_1 , so Eqn. (7) holds.

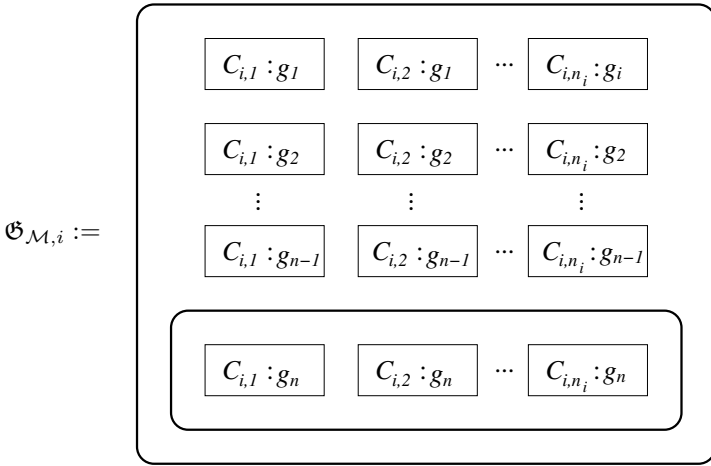
The proof of Eqn. (8) for these rules is more complex than for object implications. In contrast to object implications, we will construct for each model $\mathcal{M} \in \mathfrak{M}_2$ a graph $\mathfrak{G}_{\mathcal{M}}$ which satisfies Eqn. (8). So let $\mathcal{M} \in \mathfrak{M}_2$ be a fixed model.

Let $M := \{m_1, \dots, m_k\}$ be an enumeration of the attributes of \mathcal{M} . Due to (*), each attribute concept $\mu(m_i)$, $1 \leq i \leq k$, of \mathcal{M} is the meet of some concepts generated by concept names. We fix these concept names as follows: For each $1 \leq i \leq k$, let $C_{i,1}, \dots, C_{i,n_i}$ be n_i concept names with $\mu(m_i) = \lambda_{\mathcal{C}}(C_{i,1}) \wedge \dots \wedge \lambda_{\mathcal{C}}(C_{i,n_i})$. In order to construct a graph \mathfrak{G}_M for Eqn. (8), we transform the model restriction $\gamma(\lambda_{\mathcal{G}}(g_1)) \vee \dots \vee \gamma(\lambda_{\mathcal{G}}(g_{n-1})) \geq \gamma(\lambda_{\mathcal{G}}(g_n))$ as follows:

$$\begin{aligned} & \gamma(\lambda_{\mathcal{G}}(g_1)) \vee \dots \vee \gamma(\lambda_{\mathcal{G}}(g_{n-1})) \geq \gamma(\lambda_{\mathcal{G}}(g_n)) \\ \iff & \lambda_{\mathcal{G}}(g_1)^I \cap \dots \cap \lambda_{\mathcal{G}}(g_{n-1})^I \subseteq \lambda_{\mathcal{G}}(g_n)^I \\ \iff & \forall m \in \mathcal{M} : \lambda_{\mathcal{G}}(g_1)Im \wedge \dots \wedge \lambda_{\mathcal{G}}(g_{n-1})Im \Rightarrow \lambda_{\mathcal{G}}(g_n)Im \\ \iff & \forall m \in \mathcal{M} : \lambda_{\mathcal{G}}(g_1) \in \text{Ext}(\mu(m)) \wedge \dots \wedge \lambda_{\mathcal{G}}(g_{n-1}) \in \text{Ext}(\mu(m)) \Rightarrow \\ & \lambda_{\mathcal{G}}(g_n) \in \text{Ext}(\mu(m)) \end{aligned}$$

$$\begin{aligned}
&\iff \forall 1 \leq i \leq k : \lambda_{\mathcal{G}}(g_1) \in \text{Ext}[\lambda_{\mathcal{C}}(C_{i,1}) \wedge \dots \wedge \lambda_{\mathcal{C}}(C_{i,n_i})] \wedge \dots \wedge \\
&\quad \lambda_{\mathcal{G}}(g_{n-1}) \in \text{Ext}[\lambda_{\mathcal{C}}(C_{i,1}) \wedge \dots \wedge \lambda_{\mathcal{C}}(C_{i,n_i})] \Rightarrow \\
&\quad \lambda_{\mathcal{G}}(g_n) \in \text{Ext}[\lambda_{\mathcal{C}}(C_{i,1}) \wedge \dots \wedge \lambda_{\mathcal{C}}(C_{i,n_i})] \\
&\iff \forall 1 \leq i \leq k : \lambda_{\mathcal{G}}(g_1) \in \text{Ext}(\lambda_{\mathcal{C}}(C_{i,1})) \wedge \dots \wedge \lambda_{\mathcal{G}}(g_1) \in \text{Ext}(\lambda_{\mathcal{C}}(C_{i,n_i})) \wedge \dots \wedge \\
&\quad \lambda_{\mathcal{G}}(g_{n-1}) \in \text{Ext}(\lambda_{\mathcal{C}}(C_{i,1})) \wedge \dots \wedge \lambda_{\mathcal{G}}(g_{n-1}) \in \text{Ext}(\lambda_{\mathcal{C}}(C_{i,n_i})) \Rightarrow \\
&\quad \lambda_{\mathcal{G}}(g_n) \in \text{Ext}(\lambda_{\mathcal{C}}(C_{i,1})) \wedge \dots \wedge \lambda_{\mathcal{G}}(g_n) \in \text{Ext}(\lambda_{\mathcal{C}}(C_{i,n_i})) \quad (11)
\end{aligned}$$

Recall that $\lambda_{\mathcal{G}}(g) \in \text{Ext}(\lambda_{\mathcal{C}}(C))$ is, as a concept graph, expressed by the concept box $\boxed{C : g}$. So we see that the last equation (which depends on i) can be represented by a concept graph as follows:



It is easy to see that each graph $\mathfrak{G}_{\mathcal{M},i}$ can be derived from \vdash_2 . Let $\mathfrak{G}_{\mathcal{M}}$ be the juxtaposition of all \mathfrak{G}_i , $1 \leq i \leq k$, (which can be derived as well).

Similar to concept implications, Eqn. (11) yields that $\mathcal{M} \not\models \mathfrak{G}_{\mathcal{M}}$, so Eqn. (8) is satisfied. Again Thm. 1 yields that \vdash_2 is complete.

5 Discussion of the Results and Further Research

It has been the goal of the paper to show that object- and concept implications, which Wille introduced to semantical concept graphs, can adequately be handled in syntactical concept graphs as well. At a first glance, due to the proven results, the goal of the paper is fulfilled. A closer observation shows valuable insights into the different structure of semantical and syntactical concept graphs.

We have started with CGwCs, without any background knowledge. Then we argued that the incorporation of background knowledge corresponds to a restrictions of our models. The basic idea of this paper is that these restrictions can already be “described” with our starting CGwCs, which has been captured by Eqn. (8). Satisfying Eqn. (8) was possible because the expressive power of

CGwCs is strong enough. In contrast to the syntactical approach, for the semantical approach to concept graphs, background knowledge was introduced to concept graphs in which we have no possibility to express negation. Due to the weak expressive power of those graphs, the restrictions of our models cannot be described with syntactical concept graphs *without* cuts. Thus, the idea of this paper cannot be adopted to incorporate background knowledge to syntactical concept graphs without cuts. So the question arises how background knowledge can be handled in these graphs.

In [2,3], it is shown that the calculus for concept graph with cuts is still complete if it is restricted to the class of concept graphs *without* cuts. The proof relies on completely different methods than the proofs in this paper: It is based on so-called standard-models and standard-graphs. It is likely that the methods of [2,3] can be adopted to incorporate background knowledge to syntactical concept graphs without cuts, but of course, this has to be worked out in a precise manner.

The next remarkable fact is that the handling of object implications had been significant more complex than the handling of concept implications: We needed the further restriction (*) on page 167. For concept implications, we did not need a similar restriction. The main reason for this is the fact that -similar to first order logic- we can express with CGwCs an universal quantification for objects (in other words: we have statements which range over the set of objects). But we cannot express an universal quantification for concepts or relations⁴. In Formal Concept Analysis, thus in the semantical approach for concept graphs, the situation is different: Here we have the well-known duality for objects and attributes. We have argued that concept implications can be reduced to attribute implications, thus, due to this duality principle, it is obvious that -for the semantical approach- concept implications and attribute implications can be treated similar.

The different treatment of object- and concept implication in the syntactical approach would not arise if we used the “classical” relational models of first order logic (instead of the contextual models) as semantics for concept graphs. Remember that a relational model is a pair (U, I) , consisting of an universe of discourse U , and an interpretation function I which maps object names to objects and relation names to (extensional) relations. Particularly, we do not have more relations in this model than relation names in the underlying alphabet. For contextual models, the situation is different: We may have many more concepts or relations than we can describe with our language of syntactical concept graphs over a fixed alphabet. For this reason we had to introduce restriction (*). If (*) is satisfied, then, roughly speaking, each concept or relation of a contextual model can be described within our language of CGwCs, even if the concept or relation has no name.

In practice, the restriction (*) should have no considerable effects. This can easily be seen if we examine how Toscana-Systems are set up. This process usu-

⁴ With an universal quantification for concepts we would have the full expressive power of second order logic. But then, due to the theorems of Lindström, we could not have a sound and complete calculus.

ally starts with fixing a set of objects and multi-valued attributes, from which –usually with the process of scaling– a formal context and its concepts is derived. The crucial point for our consideration is that the knowledge experts determine the objects, attributes and the incidence relations of the context, then, afterwards, the concept lattice is derived and examined. Particularly, a knowledge experts explicates only attributes and attribute concepts, but no further concepts. Of course, these explicated attribute concepts are \wedge -dense in the set of all concepts of the formal context, i.e., (*) will obviously be fulfilled. In this sense, for incorporating the results of this paper in the development of Toscana-systems, we probably do not have to take care of (*). Nonetheless, the impact of (*) to models has to be further investigated.

References

1. R. B. Brandom: *Making it explicit. Reasoning, representing, and discursive commitment*. Harvard University Press, Cambridge 1994. Suhrkamp, 2001.
2. F. Dau: *The Logic System of Concept Graphs with Negation (And Its Relationship to Predicate Logic)*. Lecture Notes in Artificial Intelligence, Vol. 2892, ISBN 3-540-20607-8. Springer, Berlin–Heidelberg–New York, 2003.
3. F. Dau: *Concept Graphs without Negations: Standardmodels and Standardgraphs*. In: Moor, A., Lex, W., Ganter, B. (Eds.): *Conceptual Structures for Knowledge Creation and Communication*. LNAI 2746, ISBN 3-540-40576-3, Springer-Verlag, Heidelberg-Berlin, 2003, 243 pp. (This paper is a part of [2] as well.)
4. F. Dau, J. Klinger: *From Formal Concept Analysis to Contextual Logic*. To appear in the proceedings of the First International Conference on Formal Concept Analysis, 2003.
5. B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin–Heidelberg–New York 1999.
6. J. Klinger: *Semiconcept Graphs: Syntax and Semantics*. Diploma thesis, TU Darmstadt, 2001.
7. J. Klinger: *Simple Semiconcept Graphs: A Boolean Logic Approach*. In: H.S. Delugach, G. Stumme, (Eds.): *Conceptual Structures: Broadening the Base*. LNAI 2120, Springer-Verlag, Berlin–Heidelberg, 2001.
8. J. Klinger: *Semiconcept Graphs with Variables*. In G. Angelova, D. Corbett, U. Priss (Eds.): *Conceptual Structures: Integration and Interfaces*. LNAI 2393, Springer Verlag, Berlin–Heidelberg, 2002.
9. M. Nicholson: *The pocket Aussie fact book*. Penguin Books Australia, Ringwood 1999.
10. S. Pollandt: *Relational Constructions on Semiconcept Graphs*. In: B. Ganter, G. Mineau (Eds.): *Conceptual Structures: Extracting and Representing Semantics*. Contributions to ICCS 2001, Stanford.
11. S. Pollandt: *Relation Graphs: A Structure for Representing Relations in Contextual Logic of Relations*. In G. Angelova, D. Corbett, U. Priss (Eds.): *Conceptual Structures: Integration and Interfaces*. LNAI 2393, Springer Verlag, Berlin–Heidelberg 2002.
12. S. Prediger: *Kontextuelle Urteilslogik mit Begriffsgraphen. Ein Beitrag zur Restrukturierung der mathematischen Logik*. Shaker Verlag 1998.

13. S. Prediger: *Simple Concept Graphs: A Logic Approach*. In: M. -L. Mugnier, M. Chein (Eds.): *Conceptual Structures: Theory, Tools and Applications*. LNAI 1453, Springer Verlag, Berlin–New York 1998, 225–239.
14. L. Schoolmann, R. Wille: *Concept Graphs with Subdivision: A Semantic Approach*. In: A. d. Moor, W. Lex, B. Ganter (Eds.): *Conceptual Structures for Knowledge Creation and Communication*, LNAI, Springer Verlag, Berlin–Heidelberg 2003.
15. J. F. Sowa: *Conceptual Structures: Information Processing in Mind and Machine*. Addison Wesley Publishing Company Reading, 1984.
16. J. F. Sowa: *Conceptual Graphs Summary*. In: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (Eds.): *Conceptual Structures: current research and practice*, Ellis Horwood, 1992, 3–51.
17. J. F. Sowa: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.
18. R. Wille: *Plädoyer für eine Philosophische Grundlegung der Begrifflichen Wissensverarbeitung*. In: R. Wille, M. Zickwolff (Eds.): *Begriffliche Wissensverarbeitung: Grundfragen und Aufgaben*. B.I.–Wissenschaftsverlag, Mannheim, 1994, 11–25.
19. R. Wille: *Restructuring Mathematical Logic: An Approach Based on Peirce's Pragmatism*. In: A. Ursini, P. Agliano (Eds.): *Logic and Algebra*. Marcel Dekker, New York 1996, 267–281.
20. R. Wille: *Conceptual Graphs and Formal Concept Analysis*. In: D. Lukose et al. (Eds.): *Conceptual Structures: Fulfilling Peirce's Dream*. LNAI 1257, Springer Verlag, Berlin–New York 1997, 290–303. y
21. R. Wille: *Contextual Logic Summary*. In: G. Stumme (Ed.): *Working with Conceptual Structures*. Contributions to ICCS 2000. Shaker, Aachen 2000, 265–276.
22. R. Wille: *Lecture Notes on Contextual Logic of Relations*. FB4-Preprint, TU-Darmstadt, 2000.
23. R. Wille: *Existential Concept Graphs of Power Context Families*. In G. Angelova, D. Corbett, U: Priss (Eds.): *Conceptual Structures: Integration and Interfaces*. LNAI 2393, Springer Verlag, Berlin–Heidelberg 2002.
24. R. Wille: *Conceptual Contents as Information – Basics for Contextual Judgement Logic*. In: A. d. Moor, W. Lex, B. Ganter (Eds.): *Conceptual Structures for Knowledge Creation and Communication*, LNAI, Springer Verlag, Berlin–Heidelberg 2003.

Agreement Contexts in Formal Concept Analysis

Richard Cole and Peter Becker

School of Information Technology and Electrical Engineering (ITEE)

The University of Queensland

QLD 4072, Australia

{rcole,pbecker}@itee.uq.edu.au

Abstract. This paper describes a technique in which some columns of an n -ary relation are interpreted as defining a *situation*. For example when considering movies, critics and reviews we talk about the situation when the critic is a particular critic. We then construct a formal context called an *agreement context* designed to show that which is in common between the situations. We elaborate this idea in two ways: (i) by combining different types of situation; and (ii) using conceptual scales to introduce intervals of agreement.

1 Introduction

Formal concept analysis as a technique of data analysis has a long and successful history of application in various domains. Through these applications, and in concert with theoretical developments a work-flow of conceptual data analysis has been established that may be summarised as: (i) collect high quality, meaningful input data and organise it as a many-valued context, (ii) create a collection of conceptual scales with which to interpret the values in the many-valued context, and (iii) explore the landscape created by the application of these conceptual scales to the data in order to form and test hypotheses about the data and retrieve specific instances.

Classic conceptual data analysis requires that the input data can be organised as a many valued context. This assumes a functional dependency: a value has to be unique for any given object-attribute pair. Such a context can be represented in a form similar to a spreadsheet — each object is assigned a row, each attribute a column, and each cell may contain at most a single value.

Much of the data that we would wish to analyse though exists within relational database in a form that cannot be readily converted into the form of a many-valued context since either the functional dependency is lacking or the relation to be analysed is of higher order than three. A number of approaches have been defined that extend FCA to handle higher order relational data in various ways. Notable examples include: a triadic approach to concept analysis [WL95], multi-contexts [Wil96] and power context families [Wil97], relational graphs [Wil02], and reverse pivoting [HS01].

This paper provides a quite different approach to the analysis of high order relations by considering some columns of the relation as defining a *situation*. By situation we mean that part of the relation defines a context in which something can be true, while not necessarily being true in general. For example by considering a movie critic as defining a situation, we can ask within that situation if a particular movie is considered good

Table 1. Movie ratings taken from the “SBS Movie Show” by two reviewers, Margaret Pomeranz and David Stratton, organised as a ternary relation R_{ms}

Movie	Critic	Rating
The League Of Extraordinary Gentlemen	Margaret	2.5
The League Of Extraordinary Gentlemen	David	1
Matchstick Men	Margaret	4.5
Matchstick Men	David	4.5
The Wannabes	Margaret	2
The Wannabes	David	3.5
Swimming Pool	Margaret	4
Swimming Pool	David	3.5
American Splendor	Margaret	4.5
American Splendor	David	4
28 Days Later	Margaret	4
28 Days Later	David	3
Bad Boys II	Margaret	2
Bad Boys II	David	3
Japanese Story	Margaret	4.5
Japanese Story	David	4.5

or not. This notion of situation is similar in some ways to a situation within situation theory [Bar89], or a local logic within the logic of distributed systems [BS97].

Throughout the paper we will use ratings of movies by two reviewers, Margaret Pomeranz and David Stratton, from a TV show called “The Movie Show”¹. Table 1 shows the eight movies we will use in our first example and refer to as R_{ms} , other examples use an extended set of 35 movies whose reviews were conducted around the same time which we refer to as R_{ml} .

The rest of the paper is structured as follows: Section 2 introduces the notion of an *agreement context* and demonstrates its use via an example. using the data from Table 1. Section 3 explains how the notion of a agreement contexts may be combined with conceptual scaling, first as a mechanism to scale the output of the agreement context and then secondly as a way to scale relation before looking for agreement. Section 4 provides a conclusion and outlook.

2 Agreement Contexts

In order to understand the basic intuition behind agreement contexts, consider the columns *movie*, *critic* and *rating* of the example introduced in Section 1. The data contained in these columns can be seen as a relation, $R \subseteq D_1 \times D_2 \times D_3$, in which the dimensions, D_1, \dots, D_3 cover movies, critics, and ratings respectively.

Now consider what is in common between the ratings assigned by a group of critics, $S \subseteq D_2$. A movie rating pair $(m, r) \in D_1 \times D_3$ is in common between critics $s_1, s_2 \in S$ if $(m, s_1, r) \in R$ and $(m, s_2, r) \in R$. The following approach considers each critic as defining

¹ <http://www.sbs.com.au/movieshow/>

a situation and then generates a concept lattice containing the agreement between each subset of the defined situations.

First we introduce two relational projections:

Definition 1. *Given an n -ary relation R , let π_i^n and $\bar{\pi}_i^n$ be operations on tuples of R defined as:*

$$\begin{aligned}\pi_i^n(g_1, \dots, g_n) &= (g_i) \\ \bar{\pi}_i^n(g_1, \dots, g_n) &= (g_1, \dots, g_{i-1}, g_{i+1}, \dots, g_n)\end{aligned}$$

Likewise define π_i^n and $\bar{\pi}_i^n$ as operations on n -ary relation given by:

$$\pi_i^n R = \{\pi_i^n g \mid g \in R\} \quad \text{and} \quad \bar{\pi}_i^n R = \{\bar{\pi}_i^n g \mid g \in R\}$$

Given these definitions we can generalize the idea above to create a formal context to express agreement.

Definition 2. *For any relation R , we define $\mathbb{K}_{R,i}$ as:*

$$\mathbb{K}_{R,i} := (\bar{\pi}_i^n R, \pi_i^n R, I_i)$$

where

$$(g, m) \in I_i :\Leftrightarrow \exists t \in R : g = \bar{\pi}_i^n t \text{ and } m = \pi_i^n t$$

This structure is not suited for apposition, an operation on contexts needed to apply nesting and zooming as used in programs such as TOSCANAJ [BH04] and CEM [CS00]. Given $\mathbb{K}_i = (G, M_i, I_i)$, for $i \in 1, 2$ the apposition, $\mathbb{K}_1 | \mathbb{K}_2$, is defined [GW99] as:

$$\mathbb{K}_1 | \mathbb{K}_2 := (G, \dot{M}_1 \cup \dot{M}_2, \dot{I}_1 \cup \dot{I}_2)$$

where $\dot{M}_i = \{i\} \times M_i$ and $\dot{I}_i = \{(g, (i, m)) \in G \times M_i \mid (g, m) \in I_i\}$ for $i \in 1, 2$.

In order to construct an apposition, both contexts must share a common object set, and this does not hold for the family of contexts $(\mathbb{K}_{R,i})_{i=1, \dots, n}$. To obtain a common object set we introduce the following definition which uses the tuples of R as objects:

Definition 3. *The i -th agreement context of the relation R is the formal context $\mathcal{M}(R, i) := (R, \pi_i^n R, \nabla_i R)$ where*

$$\nabla_i R := \{(g, m) \in R \times \pi_i^n R \mid \exists h \in R : \bar{\pi}_i^n h = \bar{\pi}_i^n g \text{ and } \pi_i^n h = m\}$$

The concept lattice of the agreement context $\mathcal{M}(R, i)$ is isomorphic to the concept lattice of the context $\mathbb{K}_{R,i}$ since $\bar{\pi}_i^n$ is an intent preserving surjective mapping from the objects of $\mathcal{M}(R, i)$ to the objects of $\mathbb{K}_{R,i}$. Since the object set for all agreement contexts of one relation is the same, apposition can be applied and thus nesting and zooming is possible.

Example: A agreement context was generated from the data in Table 1 by considering the critic field as a situation and the concept lattice of this context is shown in Fig. 1. The concept lattice contains four concepts, the bottom concept has as extent tuples in which the ratings are common between Margaret and David.

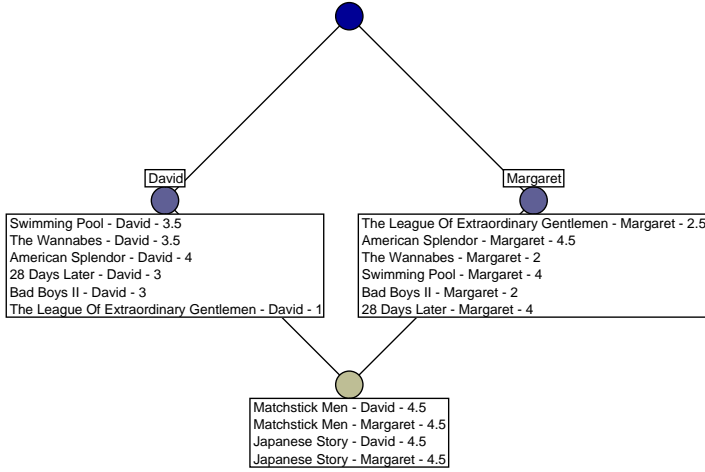


Fig. 1. Lattice for the agreement context $\mathcal{M}(R_{ms}, 2)$

The extent for the concept labeled Margaret contains (i) those reviews particular to Margaret, (ii) the reviews of Margaret that match a review of David, and (iii) the reviews of David that match a review of Margaret. The concept labeled David similarly contains those reviews that match Margaret’s, those of Margaret that match one of David’s, and those particular to David.

Two entries occurs for each movie, one for each critic in Fig. 1. As we explained earlier it is possible to recover $\mathbb{K}_{R,i}$ by applying the projection π_i^j to each extent. In our example this corresponds to removing the critics from the object labels and then removing duplicate labels. If this were done the bottom concept would contain just “Matchstick Men - 4.5” and “Japanese Story - 4.5”.

3 Agreement Contexts and Conceptual Scaling

Conceptual scaling means defining a formal context by which values from some domain may be interpreted. As an example consider the commonly used interordinal scale, I_n . This scale is used to interpret numerical values in a certain range by constructing intervals between n selected boundaries.

Many-valued contexts are defined in [GW89] as a tuple (G, M, W, I) where G , M , and W are sets, and $I \subseteq G \times M \times W$ is a relation such that w is functionally dependent on g and m , i.e. $(g, m, w_1), (g, m, w_2) \in I$ implies $w_1 = w_2$. A conceptual scale, for an attribute m is a formal context (G_s, M_s, I_s) with $G_s \supseteq \{w \in W \mid (g, m) \in I\}$. The conceptual scale can be applied to the many valued context to derive a formal context (G, M_s, J) , where $(g, m) \in J :\Leftrightarrow \exists w \in W : (g, m, w) \in I, (w, m) \in I_s$.

In the following we will describe two approaches of combining conceptual scaling with agreement contexts: first using conceptual scaling as means of visualising the agreement extents and secondly applying conceptual scaling to a relation before the agreement operations to find agreement induced by the scale.

3.1 Applying Conceptual Scales to Agreement Extents

Definition 4. Given a relation $R \subseteq D_1 \times \dots \times D_n$, and a conceptual scale $\mathbb{S} := (G_s, M_s, I_s)$, with $G_s \supseteq D_i$ the context derived with respect to agreement scaling is (R, M_s, J) where

$$(g, m) \in J : \Leftrightarrow \exists w \in D_i : (g, w) \in \nabla_i R, \text{ and } (w, m) \in I_s$$

Standard conceptual scaling relies on the functional dependency of the many-valued context in order to achieve the result that the context derived with respect to plain scaling can be \vee -embedded into the scale lattice. Definition 4 drops this requirement and thus we need to make use of a construction introduced by Stumme and Hereth [HS01], namely the power scale.

Definition 5. A power scale of a conceptual scale (G_s, M_s, I_s) , is the formal context $(\mathfrak{P}(G_s), M_s, I_s^{\mathfrak{P}})$, where

$$(A, m) \in I_s^{\mathfrak{P}} : \Leftrightarrow \exists g \in A : (g, m) \in I_s$$

for $A \subseteq G_s$, and $m \in M$.

The concept lattice of a context derived with respect to agreement scaling from a relation R and a conceptual scale \mathbb{S} can be \vee -embedded into the concept lattice of the power scale of \mathbb{S} , and if $\nabla_i R$ defines a functional relationship then the derived context can be \vee -embedded into $\mathfrak{B}(\mathbb{S})$, thus agreement scaling can be consistently combined with conceptual scaling in order to provide an interpretation for the domain to which it is applied.

Example: The example in Fig. 2 is derived from our “SBS Movie Show” example using agreement scaling with two conceptual scales. The outer scale is a boolean scale for Margaret and David, while the inner scale an interordinal scale I_6 . Since the rating is functionally dependent on the movie and the reviewer we can use I_6 rather than having to resort to the power scale of I_6 .

The bottom concept of the outer scale contains the movies whose ratings were in common between the two reviewers. For instance both reviewers agreed that “Legally Blonde 2” should have a rating of 2 stars. Similarly both critics agreed that “Matchstick Men” should deserved a review of 4.5 stars. A limitation of this approach is that the bottom outer concept contains only movie ratings that are in exact agreement between the two critics. This can be seen by considering the concepts for Margaret and David. Both critics gave low scores to Take Away — David gave it 1 while Margaret gave it 0.5. Although these ratings are close they do not result in an agreement in this structure. The approach outlined in the next section attempts to remedy this.

3.2 Finding the Agreement in a Conceptual Scale

The approach presented so far shows exact agreement and does not account for how similar two ratings may be. In the example used one might ask a question like: “What is the range of ratings given for a particular movie?”. This can be achieved by first applying the conceptual scale to a relation and then creating the agreement context.

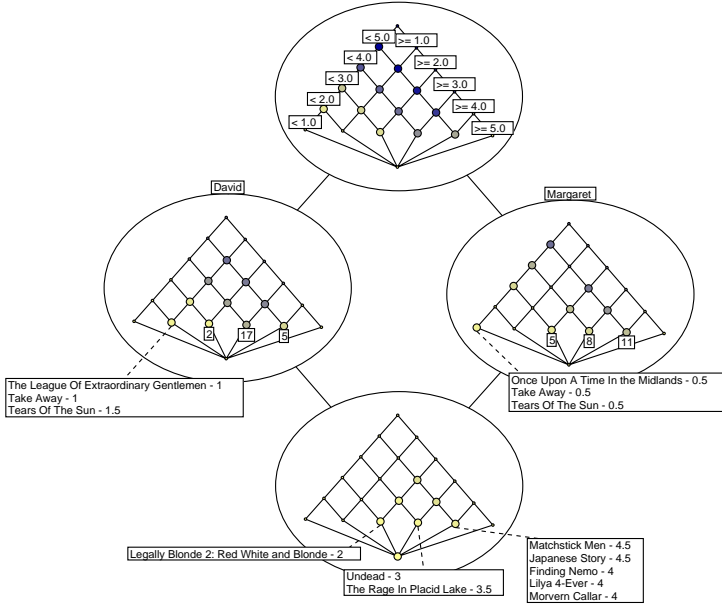


Fig. 2. A nesting of two conceptual scales of the B_2 and I_6 type, applied via agreement scaling to the second and third column of R_{ml} . The object labels contain contingents projected through $\bar{\pi}_2^3$

Definition 6. Given an n -ary relation $R \subseteq D_1 \times \dots \times D_n$ and a conceptual scale $\mathbb{S} := (G_s, M_s, I_s)$, with $G_s \supseteq D_i$ the relation derived with respect to plain scaling is:

$$R \otimes_i \mathbb{S} := \{g \in \times_{k=1}^{i-1} D_k \times M_s \times \times_{k=i+1}^n D_k \mid \exists h \in R : \bar{\pi}_n^i h = \bar{\pi}_n^i g \text{ and } (\pi_n^i h, \pi_n^i g) \in I_s\}$$

In terms of relational algebra this is a join of R with I_s : the i -th column of the relation gets replaced with the attributes of the conceptual scale. Creating agreement contexts for this relation finds agreement based on the attributes in the conceptual scale, rather than exact matches within the column begin scaled. By using different conceptual scales the commonality can be modeled with different granularities and dimensions.

Example: Fig. 3 shows the commonality between Margaret and David using the interordinal scale from the example above, but this time scaling is performed prior to constructing the agreement context.

The first step to create this diagram is to conceptually scale R_{ml} by using the same I_6 we used before. From this scaled relation we construct the agreement context and select the bottom concept whose extent corresponds to the reviews in common between Margaret and David. We generate the 3-rd agreement context which corresponds to considering the scale attributes as defining an agreement context. The resulting concept lattice is shown in Figure 3 with the following modification: We project out both the rating and the critic from the concept extents. This modification is valid since the critics are already determined to be both Margaret and David for each movie rating pair, and the ratings for each movie are specified by the concept intent.

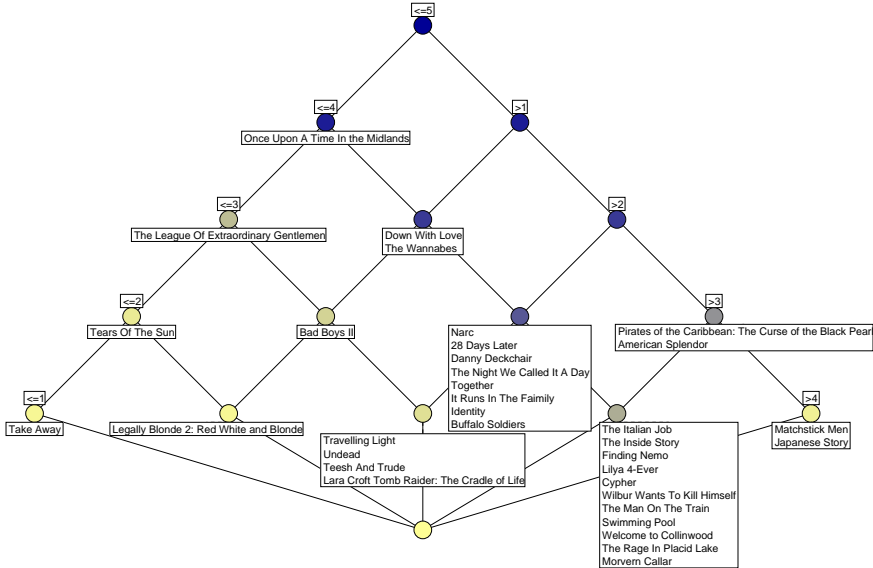


Fig. 3. Ratings in common between Margaret’s and David’s generated from the extent of the bottom concept of $\mathfrak{B}(\mathcal{M}(R_{ml} \otimes_3 I_6, 2))$. From the extent of this bottom concept an agreement context with respect to third column — the scale attributes $\leq 1, \leq 2$ etc. — has been constructed. It’s concept lattice is shown with extents projected to contain only movie titles

Note that the result is structurally an I_5 even though the scaling was done with an I_6 . This occurred because the lattice was generated using a data driven process where unrealised concepts are not drawn.

The diagram in Fig. 3 gives a good overview of the interval of agreement between the two reviewers for each movie. The movies at the widest layer of the lattice correspond to total agreement between the reviewers within the given scale. For movies in this layer the reviews can vary by at most in half a point since the narrowest interval in the scale is one point wide. The higher a movie is in this diagram, the less agreement there is. “Once Upon A Time In The Midlands” has the least agreement, the two critics agree that its rating is less than or equal to 4 points, but nothing more — Margaret gave half a point, David three and a half.

4 Conclusion and Outlook

For the purposes of experimenting with the approach we wrote a prototype called TUPLEWARE². This program allows to import relational data from different sources, such as text files, relational databases, RDF files or command line calls to other storage systems. The results can be exported to TOSCANAJ and thus nesting and zooming are supported in a limited fashion.

² <http://tockit.sourceforge.net/tupleware>

We have given an approach of interpreting dimensions of a relation as a situation and then exploring the agreement between these situations using Formal Concept Analysis. With the prototype we implemented we showed that the approach is suited to the analysis of agreement between two movie critics and that the resulting diagrams visualize different aspects of the agreement in an intuitive fashion.

The work outlined in this paper can be extended in different ways. In terms of problem domains more problems can be approached, such as comparing results from different sensors or analysing larger data sets with more or wider dimensions. In terms of the mathematisation the question of how functional dependencies between columns of the relation affect the suitability of columns to be selected as defining situations is yet to be fully explored. In terms of tool support the current prototype is still quite limited in its support for the features presented in this paper and in terms of user guidance.

References

- [Bar89] J. Barwise. *The situation in logic*. Center for the Study of Language and Information, Stanford, CA, 1989.
- [BH04] P. Becker and J. Hereth Correia. The ToscanaJ suite for implementing Conceptual Information Systems. In *Formal Concept Analysis – State of the Art*, Berlin – Heidelberg – New York, 2004. Springer. To appear.
- [BS97] J. Barwise and J. Seligman. *Information flow : the logic of distributed systems*. Cambridge University Press, Cambridge ; New York, 1997.
- [CS00] R. Cole and G. Stumme. CEM: A Conceptual Email Manager. In B. Ganter and G. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues*, number 1867 in LNAI, pages 438–452. Springer Verlag, Berlin – Heidelberg – New York, 2000.
- [GW89] B. Ganter and R. Wille. Conceptual scaling. In F. Roberts, editor, *Applications of combinatorics and graph theory to the biological and social sciences*, pages 139–167. Springer-Verlag, New York, 1989.
- [GW99] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.
- [HS01] J. Hereth and G. Stumme. Reverse pivoting in conceptual information systems. In Harry S. Delugach and Gerd Stumme, editors, *Conceptual Structures: Broadening the Base*, volume 2120 of LNAI, pages 202–215. Springer, Berlin – Heidelberg – New York, 2001.
- [Wil96] R. Wille. Conceptual structures of multi-contexts. In P. Eklund, G. Ellis, and G. Mann, editors, *Conceptual Structures: Knowledge Representation as Interlingua*, number 1114 in LNAI, pages 23–39. Springer Verlag, Berlin, 1996.
- [Wil97] R. Wille. Conceptual graphs and formal concept analysis. In D. Lukose, H. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *Conceptual structures: Fulfilling Peirce’s dream.*, volume 1257 of LNAI, pages 317–331. Springer, Berlin – Heidelberg – New York, 1997.
- [Wil02] R. Wille. Relational graphs: A structure for representing relations. In U. Priss, D. Corbett, and G. Anagelova, editors, *Conceptual Structures: Integration and Interfaces*, volume 2393 of LNAI, pages 34–47. Springer, Berlin – Heidelberg – New York, 2002.
- [WL95] R. Wille and F. Lehmann. A triadic approach to formal concept analysis. In Gerard Ellis, R. Levinson, W. Rich, and J. Sowa, editors, *Conceptual structures: applications, implementation and theory*, volume 954 of LNAI, pages 32–43. Springer, Berlin – Heidelberg – New York, 1995.

Towards a Conceptual Theory of Indistinguishable Objects

Karl Erich Wolff

Mathematics and Science Faculty
Darmstadt University of Applied Sciences
Schoefferstr. 3, D-64295 Darmstadt, Germany
karl.erich.wolff@t-online.de
<http://www.fbmn.fh-darmstadt.de/home/wolff>

Abstract. Based on the conceptual representation of objects in space and time as introduced in Temporal Concept Analysis first steps are done into the direction of a conceptual theory of “indistinguishable objects”. For that purpose a formal construction of “objects” from observed “phenomena” using their “genidentity” in conceptual transition systems is introduced. A conceptual representation of the German tale “The race between the hare and the hedgehog” is chosen as the leading example.

1 Introduction

The purpose of the paper is to start a conceptual investigation of phenomena around the term “indistinguishable objects” which occurs for example in statistics, physics, and many fields of knowledge representation. In this paper, it is impossible to give an overview over the main ideas used in the formal representation of objects and the problem of indistinguishability. Therefore, the reader is referred to the nice collection of important papers concerning the philosophical and quantum physical aspects of objects in [2]. Many of the ideas described there are closely related to the conceptual representation of objects in this paper.

1.1 Objects, Identity through Time, and Genidentity

The term “object” is used in a general way, often associated with terms like “individual”, “particle”, “subject”, or “atomic subsystem”. The conceptual representation of objects in this paper is strongly related to the problem of *identity through time* which ([2], p.7)

can also be understood in the sense of searching for “something” that unites the successive parts or momentary stages of an object. In the literature, this “something” is usually called *genidentity*, after Hans Reichenbach’s use of the term for characterizing the relation connecting different states of the same physical object at different times.

In his paper “The Genidentity of Quantum Particles” ([4], [2], p.61) Reichenbach writes

A thing is a series of events succeeding one another in time; any two events of this series are genidentical.

In the following we shall demonstrate that Reichenbach's philosophically formulated idea of "genidentity" can be described in the mathematical framework of conceptual transition systems as defined in section 4. For that purpose we use an "object construction from observed phenomena".

1.2 Object Construction from Phenomena

In previous papers on Temporal Concept Analysis [11,12,13] the author started from the usual interpretation of an object p as something which has an "identity through time". Then the idea of an "actual object (p,g) at time point g " led to the introduction of the "time relation" on the set of actual objects [12]. And that led to the definition of transitions and life tracks of objects.

In this paper we reverse the formal process from given objects to their life tracks. We now wish to investigate the process of object construction from observed phenomena. An example is the construction of the objects "Venus", "Morning Star", and "Evening Star" from some observed luminous phenomena [14]. In that paper the object construction was not yet formalized. In the actual paper we introduce a method for constructing objects from observed phenomena. These phenomena are related by a "genidentity" yielding "transitions" and "life tracks of values" which are taken as new "objects".

1.3 Indistinguishable Objects

The following investigation of indistinguishability of objects in temporal systems describes basic effects occurring when some clearly distinguished objects become indistinguishable by factoring out some information, for example deleting rows or columns of the data table of such a temporal system. If we have "time-independent" objects then equivalence relations on the set of these objects lead to the well-known Maxwell-Boltzmann, Bose-Einstein, and Fermi-Dirac statistics [4]. In this paper we focus on the question of constructing objects from given phenomena. For that purpose we now give an overview over the object representation in Temporal Concept Analysis.

2 Object Representation in Temporal Concept Analysis

To understand clearly the role of different structures in the description of objects in space and time we do not start with the usual algebraical, geometrical, topological or even metrical notions. We use a much more elementary conceptual notion which can be enriched later by classical structures.

Based on Formal Concept Analysis (FCA) introduced by Wille [5,6,3] we use the notions of Temporal Concept Analysis (TCA) as defined by the author [8,9,10,11,12,13]. The central idea in TCA is to describe temporal systems as formal contexts and "states" as object concepts of formal objects. In this paper, these formal objects are interpreted as "time points" or "time granules", but also

as “actual objects” and, more generally, as “phenomena”. The central notion of a “Conceptual Time System with Actual Objects and a Time Relation” (CTSOT) is introduced in [12]. Roughly speaking, a CTSOT consists of a scaled many-valued context divided into two parts, the time part and the event part. The formal objects of it are introduced as pairs (p, g) , called “actual objects” where p is interpreted as a “person” or an “object” and g as a “time granule” (or “time point”). The “time relation” of a CTSOT is defined as an arbitrary binary relation on the set of actual objects.

To lead the reader to a quick understanding of the main ideas we study an example of a CTSOT which describes a famous German tale with some “indistinguishable objects”.

3 The Race between the Hare and the Hedgehog

The German tale “Der Wettlauf zwischen dem Hasen und dem Igel” (The Race Between the Hare and the Hedgehog) [1] contains a nice and well-known misinterpretation on the side of the hare which is based on the fact that the hare does not distinguish between the hedgehog and its wife.

3.1 A Short Version of the Tale

At a nice Sunday morning the hedgehog leaves his family for a walk to his field where he meets the hare. They decide to make a race. The hedgehog asks his wife at home to help him in that race by waiting at the lower field. Then the hedgehog starts the race together with the hare at the upper field, but after a few steps the hedgehog stops and returns to the starting point. The hare does not realize that and runs down to the lower field where the wife of the hedgehog is waiting. Since the hare does not distinguish her from the hedgehog he thinks that the hedgehog was quicker. He repeats the race until he breaks down and dies.

3.2 The Conceptual Time System

In the following we represent the main part of the tale in a CTSOT, called \mathbf{C}_1 ; its many-valued context is explained in Table 1. This table which will be used in the following for several purposes. In the first part of the paper we do not use the row and the column labelled “phenomenon”. The CTSOT \mathbf{C}_1 has the set $P := \{\text{Hare, Hedgehog, Hedgehog's Wife}\}$ as the set of persons or objects. The set G of time granules is $G := \{0, 1, \dots, 8\}$. The set I of actual objects consists of the 26 elements in the second column; the time part has one attribute “time”, the event part has the two attributes “place” and “running”. The time relation R on the set of actual objects is defined by

$$(1) \quad (p, g) R (q, h) \iff p = q \text{ and } g < h$$

for any two objects p, q and any two time granules g, h .

The strict order relation “ $<$ ” on the set of time granules is understood here as the usual strict order relation on the set of integers.

Table 1. The many-valued context describing the main part of the race

	actual object			time	place	running
phenomenon		animal	gender	time	place	running
1	(Hedgehog,0)	hedgehog	male	0	House	no
2	(Hedgehog's Wife,0)	hedgehog	female	0	House	no
3	(Hare,1)	hare	male	1	Field	no
4	(Hedgehog,1)	hedgehog	male	1	Field	no
5	(Hedgehog's Wife,1)	hedgehog	female	1	House	no
6	(Hare,2)	hare	male	2	Field	no
7	(Hedgehog,2)	hedgehog	male	2	House	no
8	(Hedgehog's Wife,2)	hedgehog	female	2	House	no
9	(Hare,3)	hare	male	3	Upper Field	no
10	(Hedgehog,3)	hedgehog	male	3	Upper Field	no
11	(Hedgehog's Wife,3)	hedgehog	female	3	Lower Field	no
12	(Hare,4)	hare	male	4	Field	yes
13	(Hedgehog,4)	hedgehog	male	4	Field	yes
14	(Hedgehog's Wife,4)	hedgehog	female	4	Lower Field	no
15	(Hare,5)	hare	male	5	Field	yes
16	(Hedgehog,5)	hedgehog	male	5	Upper Field	no
17	(Hedgehog's Wife,5)	hedgehog	female	5	Lower Field	no
18	(Hare,6)	hare	male	6	Lower Field	no
19	(Hedgehog,6)	hedgehog	male	6	Upper Field	no
20	(Hedgehog's Wife,6)	hedgehog	female	6	Lower Field	no
21	(Hare,7)	hare	male	7	Field	yes
22	(Hedgehog,7)	hedgehog	male	7	Upper Field	no
23	(Hedgehog's Wife,7)	hedgehog	female	7	Lower Field	no
24	(Hare,8)	hare	male	8	Upper Field	no
25	(Hedgehog,8)	hedgehog	male	8	Upper Field	no
26	(Hedgehog's Wife,8)	hedgehog	female	8	Lower Field	no

The scales of the many-valued attributes are chosen as follows: an ordinal scale for the time values using the scale attributes “ ≥ 0 ” to “ ≥ 8 ”. The scales for the “place” and for “running” can be understood from Figure 1.

The bold arrows represent the life track of the hedgehog, the thin ones that of the hare, and the single dashed one the life track of the hedgehog's wife. While the concepts of “Upper Field” and “Lower Field” are sub-concepts of the concept of “Field” by the chosen scaling, the “running”-concept is also a sub-concept of it since in that context each actual object which is running is on the field.

Using CTSOTs we employ a set P of “persons” or “objects”. In the next section we reconstruct “objects” from given “phenomena”.

4 Conceptual Transition Systems

The purpose of defining Conceptual Transition Systems (CTS) is to clarify the “object construction from observed phenomena”. Therefore, we start from a

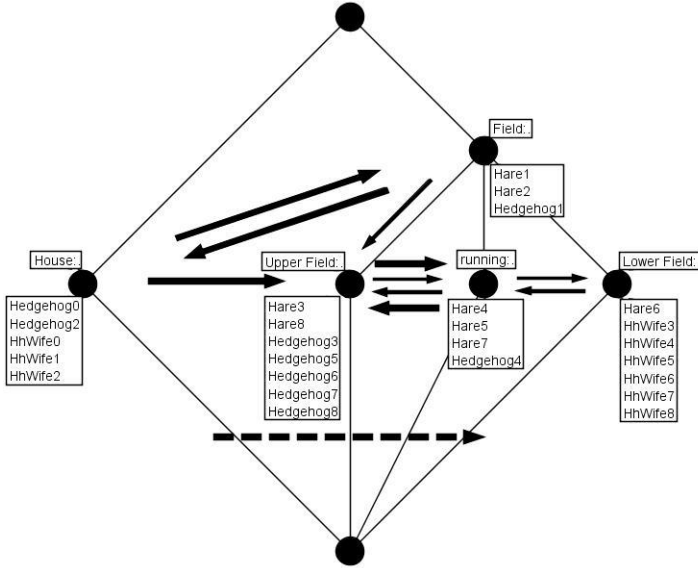


Fig. 1. The transition diagram of the “place-running” part. Hare: thin, Hedgehog: bold, Hedgehog’s Wife: dashed

many-valued context whose formal objects are interpreted as “phenomena” in the sense of “possible actual objects”. On the set of phenomena we define a “time relation” R , called the (*Reichenbach*) *genidentity* which will be used to define “life tracks” and therefrom “objects”. To represent life tracks not only in the concept lattices of the derived contexts but also in other structures as for example metric vector spaces we introduce a “frame” consisting of a set F , usually with a certain structure, and a “frame mapping” f which maps the phenomena into the chosen frame.

Definition: ‘Conceptual Transition System’

Let $\mathbf{S} := ((G, M, W, I), (S_m \mid m \in M))$ be a scaled many-valued context and let γ denote the object concept mapping of the derived context of \mathbf{S} . Let R be a binary relation on G , F a set, and $f : G \rightarrow F$ an extension of γ , that is, $f = \delta \circ \gamma$ for some mapping δ . Then the tuple (\mathbf{S}, R, f, F) is called a *conceptual transition system* (CTS), R its *genidentity*, f its *frame mapping*, and F its *frame*. If $f = \gamma$ and F is the concept lattice of the derived context, then we call the CTS a *basic* CTS and denote it by (\mathbf{S}, R) .

Examples of frames are: the direct product of the concept lattices of the scales of the CTS, the real plane (for “drawing” line diagrams), or the Euclidean affine spaces (for spatial embeddings of concept lattices).

In this paper, we need the frames mainly for defining the transition digraph in a suitable generality:

Definition: ‘R-transitions, transition digraph, transition diagram’

Let (\mathbf{S}, R, f, F) be a CTS. For any $(g, h) \in R$ the *R-transition from the start $(g, f(g))$ to the end $(h, f(h))$* is defined as the tuple $((g, h), (f(g), f(h)))$. The element $f(g)$ is called the *start-place* and $f(h)$ the *end-place* of that R-transition. Let $T(R, f)$ denote the set of all R-transitions. Then the directed multi-graph $\Gamma(R, f) := (F, T(R, f), \text{start-place, end-place})$ is called the *transition digraph* of (\mathbf{S}, R, f, F) and a graphical representation of it a *transition diagram*.

Clearly, each CTSOT can be represented as a CTS in several ways, but there is a simple construction yielding a *canonical* CTS of a given CTSOT. The main idea can be described easily in terms of a “general data table”: Replace the data table of the given CTSOT (see Table 2) by a new data table: shift the leading column of the actual objects into the “right part” of the data table by introducing two new columns “objects” and “time” and generate a new leading column for the “phenomena”. That is demonstrated in Table 3.

Table 2. General data table of a CTSOT

actual objects	time part	event part
(p, g)	v	w

Table 3. General data table of the canonical CTS of the given CTSOT

phenomena	objects	time	time part	event part
φ	p	g	v	w

The new scales for the many-valued attributes “objects” and “time” can be chosen arbitrarily. We preserve the scales of the “old” attributes. By construction, the set of the two many-valued attributes “objects” and “time” is a *key* of that many-valued context in the sense that there exists a bijection β from the set of actual objects onto the set Φ of phenomena of the CTS. We “transfer” the time relation R of the CTSOT to the genidentity R' of the canonical CTS by the definition:

$\varphi_1 R' \varphi_2 \Leftrightarrow \beta^{-1}(\varphi_1) R \beta^{-1}(\varphi_2)$. Let γ denote the object concept mapping of the derived context of the given CTSOT. Then the frame mapping f is defined by $f := \gamma \circ \beta^{-1}$ and the frame F is the image of γ .

That construction is basically used in Table 1 where we now switch from the previously described many-valued context \mathbf{C}_1 to the many-valued context \mathbf{C}_2 which has the 26 “phenomena” labelled from 1 to 26 as formal objects, and “animal”, “gender”, “time”, “place”, and “running” as attributes, and the values as given in Table 1. For example, the phenomenon 2 is described as a female hedgehog at time 0 in the house and not running, but no longer as the object “Hedgehog’s Wife”. The point is here, that we choose some “object-free” description and study how to construct “objects” from the “life tracks” generated by the genidentity on the set of “phenomena”.

4.1 The Construction of Objects

The general problem of object construction in this paper is: Given some spatiotemporal phenomena and some relations on these phenomena, construct objects in such a way that the phenomena can be explained as actual objects.

If the spatiotemporal phenomena and their relations are given by a scaled many-valued context \mathbf{S} then our basic strategy for the construction of objects is to construct a “conceptually meaningful” genidentity R leading to “meaningful” directed paths in the transition digraph of the basic CTS (\mathbf{S}, R) . These directed paths are then taken as the new objects, the “nodes” on these paths are interpreted as its “actual objects”, and they are used to construct a suitable CTSOT “explaining” the given CTS.

The mathematical details of that construction can not be given here. Instead, we demonstrate that construction of objects by applying it to the CTS which describes the point of view of the hare.

4.2 The Point of View of the Hare

From Table 1 we construct a basic CTS, called (\mathbf{S}_2, R_2) , where \mathbf{S}_2 is the scaled many-valued (“phenomena”) context in Table 1 with nominal scales for “animal” and “gender” and the same scales as in \mathbf{S}_1 for the other three attributes. R_2 is defined by $g R_2 h :\Leftrightarrow \text{animal}(g) = \text{animal}(h) \text{ and } \text{gender}(g) = \text{gender}(h) \text{ and } \text{time}(g) < \text{time}(h)$ for any two phenomena g, h . The transition diagram of (\mathbf{S}_2, R_2) yields three paths which can be interpreted as “life tracks” of three “objects”, interpreted as “Hare”, “Hedgehog”, Hedgehog’s Wife”.

In the same way we can treat the construction of the object “Hedgehog” in the point of view of the hare. Since the hare does not “see” the phenomena 1,2,5,8,11,14,16,19,23,26 we now delete these rows in Table 1 and denote the restricted set of phenomena as G_3 . We also delete the column “gender” since the hare does not distinguish between the hedgehog and its wife. The remaining scaled many-valued context is denoted by \mathbf{S}_3 . We define the genidentity R_3 by $g R_3 h :\Leftrightarrow \text{animal}(g) = \text{animal}(h) \text{ and } \text{time}(g) < \text{time}(h)$ for any two phenomena $g, h \in G_3$. The transition diagram of (\mathbf{S}_3, R_3) yields two paths which can be interpreted as “life tracks” of two “objects”, called “Hare” and “Hedgehog”. That can be seen in the transition diagram in Figure 2 where we have drawn a thin (resp. bold) transition arrow from the object concept of a phenomenon g to the object concept of the phenomenon h iff $\text{animal}(g) = \text{animal}(h) = \text{hare}$ (resp. hedgehog). Hence, we can introduce in general the notion of a “life track of a value” as for example the life track of hare resp. hedgehog. In the previous example of the CTS (\mathbf{S}_2, R_2) we have constructed the life tracks of a pair of values, for example the life track of (hedgehog, male). The mathematical description of this construction will be given elsewhere. The main idea is to replace the equality relation on the values by arbitrary relations, for example equivalence relations or order relations.

The most remarkable feature in Figure 2 is the arrow leading from the attribute concept of “running” to the attribute concept of “Lower Field” indicating the transition of the “Hedgehog” from time 4 to time 5 resulting from the R_3 -

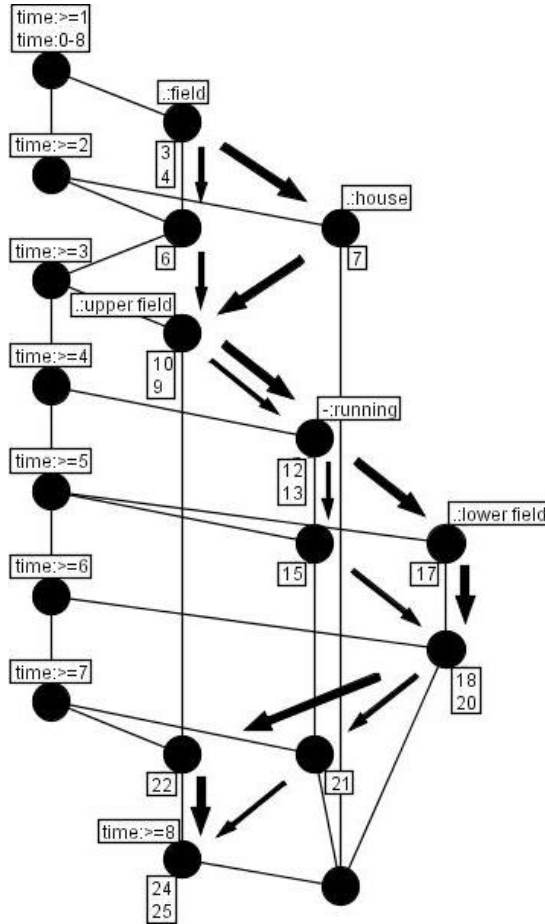


Fig. 2. The construction of the objects “Hare” (thin) and “Hedgehog” (bold) from the phenomena seen by the hare

transition (13,17) which is not an R_2 -transition, since $\text{gender}(13) = \text{male} \neq \text{female} = \text{gender}(17)$.

5 Conclusion, Applications, and Outlook

We have demonstrated the construction of temporal objects from given phenomena using “conceptual” paths in transition digraphs of Conceptual Transition Systems. That yields a useful approach for the investigation of temporal problems around the term “indistinguishable objects”. Coming from life tracks of objects we have introduced the notion of the “life track of a value”. While life tracks of objects have been used by the author in several applications [11,12,13] the more general “life tracks of values” will be useful in experimental physics for the “recognition” of particles, for the conceptual investigation of waves, and for the formal description of agents in Computer Science.

References

1. Bechstein, L.: Der Wettlauf zwischen dem Hasen und dem Igel. In: Ludwig Bechsteins Märchenbuch. F.W. Hendel Verlag Leipzig 1926, 260-264.
2. Castellani, E.(ed.): Interpreting Bodies: Classical and Quantum Objects in Modern Physics. Princeton University Press 1998.
3. Ganter, B., R. Wille: Formal Concept Analysis: mathematical foundations. (translated from the German by Cornelia Franzke) Springer-Verlag, Berlin-Heidelberg 1999.
4. Reichenbach, H.: The Direction of Time. Edited by M. Reichenbach. Berkeley: University of California Press, 1991. (Originally published in 1956)
5. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.): Ordered Sets. Reidel, Dordrecht-Boston 1982, 445-470.
6. Wille, R.: Introduction to Formal Concept Analysis. In: G. Negrini (ed.): Modelli e modellizzazione. Models and modelling. Consiglio Nazionale delle Ricerche, Istituto di Studi sulli Ricerca e Documentazione Scientifica, Roma 1997, 39-51.
7. Wolff, K.E.: A first course in Formal Concept Analysis - How to understand line diagrams. In: Faulbaum, F. (ed.): SoftStat '93, Advances in Statistical Software 4, Gustav Fischer Verlag, Stuttgart 1994, 429-438.
8. Wolff, K.E.: Concepts, States, and Systems. In: Dubois, D.M. (ed.): Computing Anticipatory Systems. CASYS'99 - Third International Conference, Liège, Belgium, 1999, American Institute of Physics, Conference Proceedings 517, 2000, pp. 83-97.
9. Wolff, K.E.: Towards a Conceptual System Theory. In: B. Sanchez, N. Nada, A. Rashid, T. Arndt, M. Sanchez (eds.): Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, SCI 2000, Vol. II: Information Systems Development, International Institute of Informatics and Systemics, 2000, 124-132.
10. Wolff, K.E.: Temporal Concept Analysis. In: E. Mephu Nguifo & al. (eds.): ICCS-2001 International Workshop on Concept Lattices-Based Theory, Methods and Tools for Knowledge Discovery in Databases, Stanford University, Palo Alto (CA), 91-107.
11. Wolff, K.E.: Transitions in Conceptual Time Systems. In: D.M.Dubois (ed.): International Journal of Computing Anticipatory Systems vol. 11, CHAOS 2002, p.398-412.
12. Wolff, K.E.: Interpretation of Automata in Temporal Concept Analysis. In: U. Priss, D. Corbett, G. Angelova (eds.): Integration and Interfaces. Tenth International Conference on Conceptual Structures. Lecture Notes in Artificial Intelligence, 2393, Springer-Verlag (2002), 341-353.
13. A Conceptual Granularity Theory for Objects in Space and Time. Preprint Darmstadt University of Applied Sciences, Mathematics and Science Faculty, 2003. To appear in the Proceedings of the First International Conference on Formal Concept Analysis, Darmstadt 2003.
14. Wolff, K.E., W. Yameogo: Time Dimension, Objects, and Life Tracks - A Conceptual Analysis. In: A. de Moor, W. Lex, B. Ganter (eds.): Conceptual Structures for Knowledge Creation and Communication. LNAI 2746, Springer. Heidelberg 2003, 188-200.

Conceptual Knowledge Processing with Formal Concept Analysis and Ontologies

Philipp Cimiano, Andreas Hotho, Gerd Stumme, and Julien Tane

Institute for Applied Informatics and Formal Description Methods (AIFB)
University of Karlsruhe, D-76128 Karlsruhe, Germany
{cimiano, hotho, stumme, tane}@aifb.uni-karlsruhe.de
www.aifb.uni-karlsruhe.de/WBS

Abstract. Among many other knowledge representations formalisms, Ontologies and Formal Concept Analysis (FCA) aim at modeling ‘concepts’. We discuss how these two formalisms may complement another from an application point of view. In particular, we will see how FCA can be used to support Ontology Engineering, and how ontologies can be exploited in FCA applications. The interplay of FCA and ontologies is studied along the life cycle of an ontology: (i) FCA can support the building of the ontology as a learning technique. (ii) The established ontology can be analyzed and navigated by using techniques of FCA. (iii) Last but not least, the ontology may be used to improve an FCA application.

1 Introduction

As concepts are the most basic units of thought, it is not surprising that they became important building blocks in Artificial Intelligence research. Their appearance is prevailing in Knowledge Representation (e. g., in semantic networks, conceptual graphs, description logics), but they also appear for instance in Machine Learning (e. g., in conceptual clustering, concept learning). All these approaches focus on other aspects of concepts, leading to different formalizations.

In this paper, we focus on two of these formalizations, namely Ontologies and Formal Concept Analysis. We will analyze how the two approaches can complement each other from an application point of view. In particular, we will discuss how Formal Concept Analysis can be used to support Ontology Engineering, and how ontologies can be exploited in FCA applications. We will here illustrate these aspects by selected applications which we have set up during the last years.

The interplay of FCA and ontologies can be seen along the life cycle of an ontology:

1. FCA can support the building of the ontology as a learning technique.
2. The established ontology can be analyzed and navigated by using techniques of FCA.
3. Last but not least, the ontology may be used to improve an FCA application.

The role of FCA in these three steps of the ontology life cycle is discussed in Sections 3 to 5, resp. Each of these sections provides two examples, and concludes with a discussion of related work. The first example in each section has been presented already sometime

ago, we recall it only briefly here to sketch the whole picture. The second example in each of the three sections is more recent, and is discussed in more depth. The discussions of related work are rather selective in that we consider only articles which address explicitly the combination of Formal Concept Analysis and ontologies. Before having a look at these applications, we will start in the next section with a more in-depth discussion of the relationship between ontologies and Formal Concept Analysis.

2 Concepts in Ontologies and in Formal Concept Analysis

‘Ontology’ in its original sense is a philosophical discipline dealing with the potentialities and conditions of being. Within Computer Science, ‘ontologies’ have been introduced about a decade ago as a means for formally representing knowledge. Following [23], they are considered as an “explicit, [formal,] specification of a [shared] conceptualization [of a domain of interest]”¹. This means that ontologies serve as representation in some pre-defined formalism of those concepts and their relationships which are needed to model a certain application domain. Three major uses of ontologies can be distinguished: communication (between machines and/or humans), automated reasoning, and representation and reuse of knowledge. Even though it seems at first glance that the use of the term ‘ontology’ differs a lot in philosophy and computer science, W. Hesse pointed out in [26] that its computer science understanding fits rather well with interpretations of last century philosophy.

As we have just seen, ontologies have the ontological status of a model. Their purpose is to model a shared understanding of the reality as perceived by some individuals in order to support knowledge-intensive applications. Formal Concept Analysis, on the other hand, plays a different role. Concept lattices are not understood as modeling some part of the reality, but rather as an artifact, which is derived from some dataset. This artifact is intended to support the user in analyzing and structuring the domain, based on the given data. While ontologies can be established without any given data, FCA relies thus always on some set of objects. Thus, in FCA, extensional and intensional aspects are equally important, while ontologies emphasize on the intensional part.

The interaction between FCA and ontologies may go in two directions. On one hand, FCA can be used as a technique for Ontology Engineering. It supports the structuring of some given data by means of concept lattices. They can be used either to extract, from a given dataset, a conceptual hierarchy which may serve as a basis for the manual or semi-automatic development of an ontology. Or they are used for visualizing the ontology, in order to support navigation and analysis tasks. For both aspects, there are basically two ways how FCA and ontology notions can be combined. The most obvious way from a theoretical viewpoint is to identify the ontology concepts with the formal concepts of FCA. In many applications, however, it turns out that the canonical match is between the ontology concepts and the FCA attributes. While FCA theory forces a distinction between concepts and attributes, that distinction is not that sharp in the ontology world. Or, as the German standard DIN 2330 states, attributes “are units of thought which are

¹ Gruber’s original version is without the words ‘formal’, ‘share’, and ‘of a domain of interest’, which nowadays are rather accepted to describe more precisely the intention of ontologies with Computer Science.

gained by abstraction, and hence they are also concepts. For building concepts, one always needs other concepts, which then play the role of attributes” (translated from [20] by the authors). In fact, the decision if something is to be modeled as an attribute or as a concept, is a discussion which is always coming up in Ontology Engineering. It is though an interesting research topic, how the dual role of attributes and concepts can be incorporated in FCA theory. In Sections 3 and 4, we will see how the two ways of matching ontology concepts with formal concepts or with FCA attributes show up in selected applications.

On the other hand, ontologies can be used to improve FCA applications. In standard Formal Concept Analysis, the set of attributes does not carry any structure. By considering this set as a set of ontology concepts, we can model relations and dependencies between the attributes. Although this does not increase the complexity of the resulting lattices (as concept lattices cover, up to isomorphism, the whole class of complete lattices), it enriches the conceptual structure and provides new means of interaction and analysis. In Section 5, we will discuss two applications of this conceptual enrichment.

Beside those two directions of interaction — which may be closed to a loop — there is also the possibility of a tighter integration. One attempt is for instance Wille et al.’s work on Contextual Logic, where FCA is considered as a theory for concepts as basic units of thought, while concept(ual) graphs are formalizing judgments and conclusions [59,42] (see also [57]). This combination of FCA and conceptual graphs aims thus at one unifying theory for formalizing traditional logic based on the triad concept – judgment – conclusion. This tighter integration, however, will not be discussed here. In the following sections we focus rather on the first two ways of interaction.

3 Ontology Learning with Formal Concept Analysis

As mentioned above, an ontology is an explicit specification of a conceptualization. However, in reality most conceptualizations are not made explicit but are rather implicit in documents, people’s heads or even in actions carried out by them. In this sense, a big challenge is to externalize the knowledge implicitly contained in these sources and to crystallize it into an ontology, i.e. a formal and explicit conceptualization.

It is also not realistic to assume that there will be one single and complete ontology for a given domain of interest. In fact, it is common in our world that different views on a certain topic or domain co-exist, each of them with their own level of detail, granularity, completeness and with their own focus. Thus, a very important issue is to combine the view of different parties on a certain domain, i.e. to merge their respective ontologies. In this section we address these two very important topics: In Section 3.1, we tackle the problem of merging existing ontologies together with the help of FCA. In Section 3.2, we focus then on the externalization of the knowledge which is implicit in texts and show how FCA can be exploited for this purpose.

3.1 Ontology Merging

The process of *ontology merging* takes as input two (or more) source ontologies and returns a merged ontology based on the given source ontologies. The resulting ontology may then be used for translating between applications which are based on their respective source ontologies. High quality results of the merging process will always

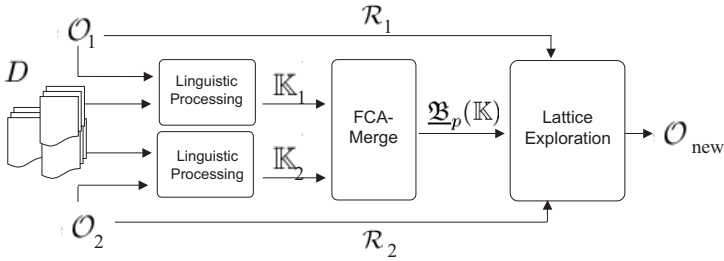


Fig. 1. The FCA-MERGE approach

need a human involved who is able to make judgments based on background knowledge, social conventions, and purposes. Thus, all merging approaches aim at supporting the knowledge engineer, and not at replacing him.

In [51], we presented the method FCA-MERGE for merging ontologies following a bottom-up approach and offering a global structural description of the merging process. For each source ontology, it extracts instances from a given set of domain-specific text documents by applying natural language processing techniques (see left part of Figure 1). This way, one context is computed for each source ontology. Its objects are the documents, and its attributes are the ontology concepts. An ontology concept is related to a document iff it occurs in the document. The contexts are joined by context apposition, and a pruned concept lattice is computed with the TITANIC algorithm [52] (see middle of Figure 1). The concept lattice provides a hierarchical, conceptual clustering of the concepts of the source ontologies. It is explored and interactively transformed to the merged ontology by the ontology engineer.

Observe that, in this approach, the ontology concepts have been identified with the FCA attributes, and not with the formal concepts. This seems to be the natural approach, as they appear already as input to the FCA step, while formal concepts only show up at the end of this step.

3.2 Ontology Learning from Text

There is no doubt that oral communication or text understanding presuppose some sort of *common ground* between the communicating partners or between the writer and the reader of a text [13]. However, this common ground or conceptualization is typically not made explicit in the conversation or the text itself. Brewster et al. [5] for example have argued that text writing and reading is in fact a process of background knowledge maintenance in the sense that basic domain knowledge is assumed, and only the relevant part of knowledge which is the issue of the article is mentioned in a more or less explicit way. Actually, knowledge can be found in texts at different levels of explicitness depending on the sort of text considered. Handbooks, textbooks or dictionaries for example contain explicit knowledge in form of definitions such as “a tiger is a mammal” or “mammals such as tigers, lions or elephants”. In fact, some researchers have exploited such regular patterns to discover taxonomic or meronymic relations in texts [24,9]. However, it seems that the more technical and specialized the texts get, the less basic knowledge we will find in them stated in an explicit way. Thus, an interesting alternative is to derive

Table 1. Object/Verb Relationships as formal context

	bookable	rentable	driveable	rideable	joinable
hotel	x				
apartment	x	x			
car	x	x	x		
bike	x	x	x	x	
excursion	x				x
trip	x				x

knowledge from texts by analyzing how certain terms are used rather than to look for explicit definitions of them. In this line the *distributional hypothesis* assumes that terms are similar to the extent to which they share similar linguistic contexts. Verbs for example give information about the state in which certain objects are or about which actions are carried out on them.

Let's for example assume that we are interested in deriving some sort of conceptualization for the tourism domain by analyzing texts related to this area. By looking at verbs as well as their direct objects we could for example derive a formal context as depicted in Table 1. If we now make the underlying assumption that the relations in Table 1 are more or less complete in the sense that we have a 'closed world' and all non-occurring relations in the text are regarded as negative examples, we could group the objects into classes or even into a concept hierarchy by analyzing their shared linguistic contexts. This is in fact the assumption of most clustering techniques which have tried to group terms appearing in texts into meaningful classes or even term hierarchies [27,3,38,6,46,4]. In this line it is an interesting option to make use of FCA to structure such terms into abstract units or concepts.

The concept lattice of the formal context in Table 1 is depicted in Figure 2. It can be transformed into a partial order as shown in the same figure in a straightforward way by removing the bottom element, introducing an ontological concept for each formal concept (named with the intent) and introducing a subconcept for each element in the contingent of the formal concept in question. In fact, this is the aim of our ongoing work presented in [11] and [12]. In order to derive the necessary verb/object dependencies from text we make use of a natural language parser, which produces a syntactic tree for each sentence in the text and from which these dependencies can be obtained in a straightforward way. Furthermore, we normalize the extracted verbs and objects by lemmatizing them via a lexicon lookup, i.e. *bought/buys* are transformed to the infinitive *buy*, and *hotels* is transformed to the singular form *hotel*. Furthermore, we also add the postfix '-able' to verbs to make them look more like attributes and to facilitate human understanding of our automatically derived lattices and concept hierarchies. In this context, there are three important issues to consider:

1. the output of the parser can be erroneous, i.e. not all derived verb/object dependencies are correct,
2. not all the derived dependencies are 'interesting' in the sense that they will help to discriminate between the different objects,

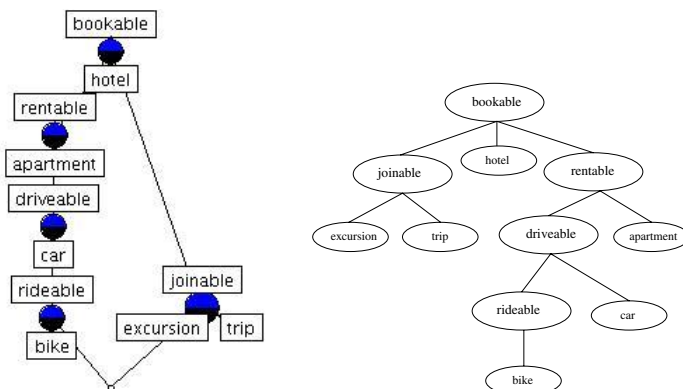


Fig. 2. The lattice and partial order for the tourism example

- the assumption of completeness of information will never be fulfilled, i.e. the text collection will never be big enough to find all the possible occurrences (compare [60]).

To deal with the first two problems, we weight the verb/object dependencies with regard to a certain information measure. In [11] we explored different measures and found out that the simple conditional probability works well enough for our purposes. The use of such an information measure deals with the first two problems as we consider only those verb/object relations for which the mentioned information measure is above some threshold t . The third problem is dealt with by our 'Smooth FCA' approach presented in [12]. In this approach we iteratively cluster terms and verbs together which are mutually most similar with respect to a certain similarity measure and thus artificially create more attribute/object pairs. The result is an overall 'smoothing' of the relative frequency landscape by assigning some non-zero relative frequencies to combinations of verbs and objects which were actually not found in the corpus. Our experiments so far have actually shown that the use of this smoothing technique actually improves the quality of the learned concept hierarchies [12]. Figure 3 shows a lattice which was automatically derived from a set of texts acquired from <http://www.lonelyplanet.com> as well as <http://www.all-in-all.de>, a webpage containing information about the history, accommodation facilities as well as activities of *Mecklenburg Vorpommern*, a region in northeast Germany. We only extracted verb/object pairs for the terms in Table 1. The corpus size was about a million words and we used the *Resnik* measure described in [11] to weight the extracted verb/object pairs. As it can be concluded from Figure 3, for *excursion* no verb/object dependencies were found in the corpus.

3.3 Related Work

The idea of using FCA in NLP in general is certainly not new. In [43] for example, several possible applications of FCA in analyzing linguistic structures, lexical semantics and lexical tuning are mentioned. [47] and [39] apply FCA to yield more concise lexical

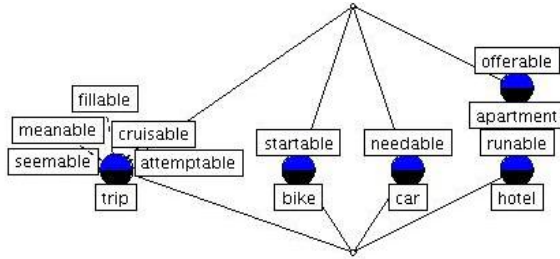


Fig. 3. The lattice automatically derived from tourism-related texts

inheritance hierarchies with regard to morphological features such as numerus, gender etc. However, to our knowledge, FCA has not been applied before to the acquisition of domain concept hierarchies such as in the approach described in this paper.

FCA-Merge is not the first tool for supporting the merging process of ontologies. The need for some support in the merging process showed up soon, since manual ontology merging using conventional editing tools is rather difficult, labor intensive and error prone. Therefore, several systems and frameworks have recently been proposed [31,8,37,35]. They rely on syntactic and semantic matching heuristics which are derived from the behavior of ontology engineers when confronted with the task of merging ontologies, i. e., human behavior is simulated. Although some of them locally use different kinds of logics for comparisons, these approaches do not offer a structural description of the global merging process.

Up to our knowledge, there is no other publication on FCA-based ontology merging, with one exception, namely OntEx, which we presented in [21]. OntEx is based on Bernhard Ganter's Attribute Exploration. It starts with the concepts of the source ontologies, and with all known hierarchical relationships. It interacts with the knowledge engineer by asking him, if further hierarchical relationships hold. If the user agrees, then the relation between the concepts is added, if he denies, then he has to provide a counterexample, which is exploited in the sequel. As FCA-Merge, OntEx also gives some guarantee that all possible combinations are considered. However, to be applicable in practice, it has to be combined with heuristic methods, as there is a price to pay for this guarantee because of the high number of interactions.

4 Navigating Ontologies with FCA

Ontologies are most often too large to be displayed on a single computer screen. Therefore, visualization and interaction techniques are needed, if the intended use of the ontology comprises a direct user access to the ontology and/or knowledge base. This is for instance the case for ontology-based web portals (see, e. g., the OntoWeb portal²). In this section, we present two applications where Formal Concept Analysis supports the navigation within an ontology-based application: the first system is an email management system, the second a management system for distributed learning repositories.

² <http://ontoweb.aifb.uni-karlsruhe.de/>

4.1 Conceptual Email Manager

The way standard email management systems store mails is directly derived from the tree structure of filing cabinets and file management systems. This has the advantage that trees have a simple structure which can easily be explained to novice users. The disadvantage is that at the moment of storing an email the user already has to foresee the way she is going to retrieve the mail later. The tree structure forces her to decide at that moment which criteria to consider as primary and which as secondary. This problem arises especially if a user communicates with overlapping communities on different topics.

In [16] (see also [15]), we presented the *Conceptual Email Manager CEM*³. It uses a simple ontology for storing the emails. The ontology consists of a hierarchy of concepts (catchwords) together with a lexicon. The hierarchy of the ontology may be any partially ordered set; multiple inheritance is thus explicitly allowed. Emails may be assigned to more than one concept, i. e., multiple instantiation is also allowed. In fact, the advantages of FCA support for navigation and retrieval tasks only comes to its full potential, when multiple inheritance and/or multiple instantiation show up.

From a FCA perspective, multiple instantiation can be seen as storing the emails in a formal context. The objects are the emails, and the attributes are the catchwords. In this setting, FCA attributes and ontology concepts are thus considered as equivalent. (Multiple) inheritance is then modeled as an additional partial order on the set of attributes, which has to be reflected by the binary relation of the context: if an email is related to a catchword, then it also has to be related to all catchwords which are higher in the partial order. This allows to assign explicitly only very specific attributes to an email (e. g., ‘ICFCA 2004’), while more general attributes are then inherited (e. g., ‘Conference submission’). Different terms in different languages can be assigned to the same attribute within the lexicon. It allows to extract the assignment of attributes to emails automatically by means of Natural Language Processing techniques, reducing thus the workload of the user⁴.

Multiple inheritance and multiple instantiation allow the user to retrieve emails via a concept lattice following different paths. This means that one does not need to decide which of the paths to use for storing. For retrieving the mail later, one can consider any combination of the catchwords. This is made possible by using the richly structured concept lattice as search space.

The ontology provides also a mechanism for automatically generating new conceptual scales. Each catchword gives rise to one conceptual scale, which has as attributes all its immediate subconcepts. The set of scales inherits the hierarchy from the ontology, so that one can focus into a single concept by selecting a next, more specific scale. This kind of ‘conceptual focusing’ is visualized by means of local scaling [49].

4.2 Courseware Watchdog

On a personal computer, it is possible to organize resources according to personal needs. In the case of remotely stored resources, this is not possible anymore, since their storage

³ A commercial follow-up of the Conceptual Email Manager is offered by Email Analysis Pty Ltd, Sydney, see <http://www.mail-sleuth.com>

⁴ If wanted, the user can still modify the result afterwards.

is not under the control of the user. Through the use of hypertext, remote material can be linked and retrieved when needed, but the particular problem of finding and organizing this remote material becomes even more crucial.

Research in the E-Learning domain shows that standards are needed for interoperability, but true interoperability does not only need data integration, it also has to consider the integration of applications. In [45], we illustrate such an integration of E-learning related applications with the implementation of a Courseware Watchdog. It is part of the PADLR project (Personalized Access to Distributed Learning Repositories) that builds upon a peer-to-peer approach for supporting personalized access to learning material⁵.

The Courseware Watchdog is built on top of the Karlsruhe Ontology and Semantic Web Framework KAON [18]⁶. KAON offers abstractions for ontologies and text corpora, an ontology editor and application framework, inferencing, and persistence mechanisms, etc. The Watchdog consists of the following components which are organized around an ontology management system:

1. *Visualization and interactive browsing techniques* allow the browsing of ontology and knowledge base in order to improve the interaction between of the user with the content.
2. A *focused crawler* finds related web sites and documents that match the user's interests. The crawl can be focused by checking new documents against the user's preferences as specified in terms of the ontology.
3. An *Edutella peer* enables querying for meta-data on learning objects with an expressive query language, and allows to publish local resources in the P2P network.
4. A *subjective clustering component* is used to generate subjective views onto the documents.
5. An *ontology evolution component* comprises ontology learning methods which discover changes and trends within the field of interest.

Here, we focus on the first component.

Ontologies are based two kinds of relations: hierarchical and non-hierarchical relations. For each kind of relation, we use an appropriate technique: the display of hierarchies through concept lattices in the first case, and relational browsing else. The browsing component is implemented in the KAON framework, and extends the Concept Explorer⁷ as a library.

We display the subsumption hierarchies of the ontologies (and in a similar way other hierarchical relations as for instance property or topic hierarchies) by embedding them into their Dedekind-McNeille completion, i. e., by displaying the concept lattice of the formal context (C, C, \leq) , where C is the set of concepts of the ontology, and \leq_C is the subsumption hierarchy. Unlike tree-oriented visualization techniques, this allows for the display of multiple inheritance. As in the Conceptual Email Manager, we also introduce instances and allow multiple instantiation. This is displayed as the concept lattice of the formal context $(C \cup I, C, \leq_C \cup \text{is-a})$ where I is the set of instances, and is-a is the instantiation relation between I and C . Using this approach, some new concepts may

⁵ <http://www.learninglab.de/english/projects/padlr.html>

⁶ <http://kaon.semanticweb.org>

⁷ <http://www.sourceforge.net/projects/conexp>

appear, even if they were not explicitly modeled within the ontology. These concepts may be used as a hint to the user to name them and to include them in his ontology.

In contrast to the Conceptual Email Manager, the Courseware Watchdog also has to consider non-hierarchical relations in the ontology. These relations represent links between diverse elements of the ontology (e. g., the “lecturer” of a “course” should be linked to it by a relation “holdsCourse”). These kinds of relation are best understood and used through some kind of exploration. Relational browsing is a technique consisting in offering the user different links which he can choose to follow. In addition to normal browsing along hyperlinks, the links are typed according to the ontology. It is possible to navigate and explore the ontology following the relations of the ontology, and then display different kinds of hierarchies.

The browsing component is also a generic way of interacting with the ontology within the other modules of the Courseware Watchdog. I. e., the user can select concepts in the Hasse diagram for use in the crawling, clustering or evolution process or for querying the P2P network.

Within the framework of Formal Concept Analysis, ontologies can be considered (under some constraints) as (many-valued) multi-contexts. We are currently investigating how to formalize this relationship, and how to exploit it for a tighter integration. In particular, we want to enhance further the interaction between FCA navigation metaphors and ontology-based knowledge management techniques.

4.3 Related Work

Systems like WAVE for web navigation [34], the expert system MCRDR [44], the RFCA system for browsing rental advertisements on the WWW [14] were first prototypes integrating both FCA and ontologies. The work on the Conceptual Email Manager was inspired by these systems. It is also related to the use of *virtual folders* in the program View Mail (VM) [32], which are collections of email documents retrieved in response to a query.

A promising next step along the line of the Conceptual Email Manager and the Courseware Watchdog will be to establish interfaces between the KAON framework and the research and software project ‘Tockit — Framework for Conceptual Knowledge Processing’⁸, in order to obtain a large, stable platform for future FCA/ontology-based projects.

5 Using Ontologies for Formal Concept Analysis Applications

One of the criteria that made Formal Concept Analysis successful is certainly the fact that, for most applications, only rather simple mathematical concepts are needed — the simplest just being a binary relation between two sets. Surprisingly many applications benefited from FCA just in this basic form. However, there were also many applications which showed the need of a more expressive knowledge representation. In this line arose for instance many-valued contexts, multicontexts, triadic contexts, and the like. The challenge for research in Formal Concept Analysis was (and still is) to bring together

⁸ <http://tockit.sourceforge.net/>

these more complex kinds of knowledge representation with the charm of the natural appearance of a (concept) lattice out of the Galois connection induced by a binary relation. In the next subsection, we will discuss one way how to derive a concept lattice out of knowledge represented in a description logic.

Another important problem in FCA applications is the size of the concept lattices. In the worst case, their size is exponential in the size of the formal context. Hence methods for managing and structuring large conceptual hierarchies and for visualizing (parts of) them have been developed. These include for instance conceptual scaling, local scaling, and iceberg concept lattices. In the second part of this section, we show a new, orthogonal approach: By summarizing ‘similar’ objects prior to the lattice computation with a standard clustering algorithm together with background knowledge encoded in an ontology, we are able to reduce significantly the size of the concept lattice without losing too much information.

5.1 Defining FCA Attributes with Description Logics

The best known approach for deriving a concept lattice from a dataset which is more than just one binary relation is conceptual scaling. It allows to derive unary attributes from many-valued ones, which are then the input to the lattice computation. However, conceptual scaling still requires that the data is represented in one (database) relation with the object name being a primary key. Conceptual scaling is thus not able to deal with more than one relation. In FCA, the problem of multiple relations has been encoded in the definition of (many-valued) multicontexts [22,58], but a theory analog to conceptual scaling, which allows the transformation of a multicontext into a meaningful structure of concept lattices has not completely worked out up to now.

However, first steps in this direction have been made. One of them is logical scaling. In [40], S. Prediger showed how a formal context (and thus a concept lattice) can be derived from a database, once a set of new attributes together with definitions in some logical language is given. In [41], we extended this approach to pre-definable conceptual scales. Logical scales can thus seamlessly be used together with ‘standard’ conceptual scales. In particular, they can be combined in nested line diagrams.

In order to illustrate the approach, we discuss a small example (see also [41]): Consider the database consisting of the two relations shown in the upper half of Figure 4. In terms of FCA it is a many-valued multicontext, in terms of Description Logics (DL) it is an A-Box. Assume that we want to classify the persons according to their wine drinking behavior. With the set of definitions shown in the lower part of Figure 4 (T-Box in DL terminology), we can define typical profiles of wine fans⁹. These definitions yield two possible scales: the data-driven and the theory-driven logical scale.

The realized data-driven scale [40] of our example is shown in the left of Figure 5. It is derived from the database by restricting the set of instances to those of the concept ‘Person’, and by taking all defined concepts of the T-Box as attributes. From the diagram, one can for instance read that all Bordeaux drinkers are also drinking red wine. However, it is not clear from the diagram if this implication holds for all possible objects (i. e., if

⁹ For a description of how to read these definitions, refer to the article of F. Baader in this volume.

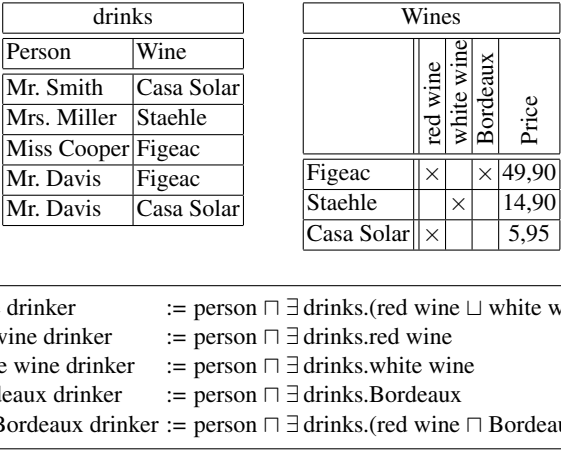


Fig. 4. A many-valued multi-context (top) and the definition of a logical scale (bottom)

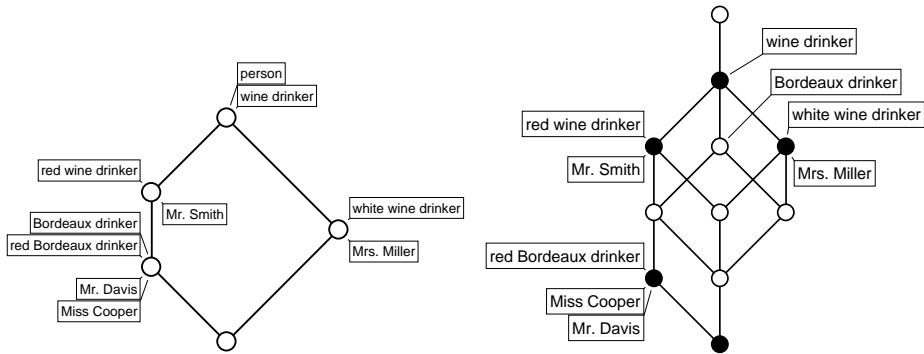


Fig. 5. The resulting data-driven (left) and theory-driven realized scales

it is forced by the definitions in the T-Box) or if it is just a coincidence because our set of instances does not cover all proto-typical cases.

Theory-driven scales [41] take this distinction into account. They consider all possible conjunctions of the defined attributes in the T-Box. The scale is computed with B. Ganter's Attribute Exploration using a DL subsumption algorithm as 'expert' (see [2] and Baaders article in this volume). The realized theory-driven scale of our example is shown in the right side of Figure 5. The data-driven scale is embedded in it as a join-semi-lattice. In the diagram, we can read the implications based on the actual set of instances as well, but it also shows which further attribute combinations are possible. For instance, one can see that the definitions do not force Bordeaux drinkers to drink red wine (they may also drink white Bordeaux only). One can also see that 'wine drinker' is the most general of the defined attributes, but that there may also exist instances not belonging to it, as this concept is different from the top concept.

The approaches presented here can be applied to any DL, provided that the instance problem is decidable within that language. I.e., one needs a sound and complete algo-

rithm which determines whether an instance from the A-Box belongs, in each model, to a given concept from the T-Box or not. If there are more than five to six definitions within the T-Box, they have to be grouped thematically into smaller subsets in order to become scales of reasonable size; each subset is giving rise to one logical scale.

5.2 Preprocessing Large Datasets for FCA

For a long time, conceptual scaling has been considered as *the* technique for dealing with large datasets in FCA: if one projects the data set only on those attributes the user is currently interested in, then the size of the lattice remains small enough to be displayed on a computer screen or on a sheet of paper. Conceptual scaling is still a very useful technique when the user has an idea which attributes (or which conceptual scales) are relevant to his task at hand. However, if the user is first confronted with a dataset, he needs some support to gain first insights, before he can start to select systematically attributes or conceptual scales.

One solution to this problem are iceberg concept lattices [52]. They can be computed without requiring any initial effort from the user, and provide a view on the top level of the concept lattice. Starting from this, the user can then continue the exploration with conventional means such as conceptual scaling (e. g., with TOSCANA [56]) or by locally analyzing the concept lattice (e. g., with the GALOIS system [7]).

Here, we will discuss another approach: If two objects are almost but not completely identical (according to the context), then they give rise to different object concepts, no matter how small the difference is. This effect can be propagated all along the concept lattice. Hence, if we allow to identify objects which are ‘almost identical’, then we may significantly reduce the size of the concept lattice, without losing too many details. The resulting — relatively small — lattice can be used as a starting point for an exploration. If one wants to analyze the data on a more detailed level, where also minor differences between objects have to be considered, one may still come back to the original concept lattice (and analyze it using conceptual scaling or local focusing [7]).

We tested this approach in a text mining scenario [30,28], where we studied the use of background knowledge in form of an ontology for clustering sets of text documents. We selected the Reuters-21578¹⁰ text collection for our experiments. The corpus consists of 21578 documents, and is especially interesting for the evaluation of clustering algorithms, as it comes along with a hand-crafted classification.

We performed the usual preprocessing steps for text mining within TextToOnto¹¹, a text mining system we developed at our institute within the KAON framework: extracting a bag-of-words model (vector space model), stopword removal, stemming, and tfidf weighting. After that, each document is represented as a vector in the vector space \mathbb{R}^n where n is the number of remaining word stems. The i th component of the vector of a document indicates the weighted frequency of the i th word stem within the document.

The vector space can also be seen as a many-valued context: each document is an object, each word stem an attribute, and the attribute values are the weighted frequencies. By conceptual scaling (e. g., by nominal scaling), one can transform this context into a

¹⁰ <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

¹¹ <http://sourceforge.net/projects/texttoonto/>

one-valued one, and can compute the concept lattice. We performed this for the Reuters dataset, but were not convinced from the results: it was very difficult to find the right values for the nominal scales. Hence the lattice was either much too detailed to provide any insight, or we obtained a flat hierarchy.

Our next step was to cluster the documents using the BiSec- k -Means algorithm before applying FCA. Instead of the individual documents, we consider now the clusters as objects of the formal context. The set of attributes remains unchanged, and the attribute values were taken from the centroid vector of each cluster. This clustering step is a generalization of the FCA task of purifying the object set of the context, in that it does not only consider completely identical objects (in the sense of the context) to be identified, but also ‘almost identical’ ones. Especially in text mining this makes sense, as almost no two documents have exactly the same representation in the bag-of-words model.

While this pre-clustering provided concept lattices that were smaller than the ones obtained by conceptual scaling, and were thus easier to interpret, they still suffered from the problem that they were rather flat. The problem is that documents may be closely related even though they use different expressions, which cannot be identified by pure syntactical methods. For instance, one document may talk about ‘beef’ and another one about ‘pork’, but beef and pork will be considered as unrelated dimensions in the vector space. This is now where background knowledge in form of an ontology comes into play: if we know that both ‘beef’ and ‘pork’ are subsumed under ‘meat’, then the cluster algorithm can assign both documents to the same cluster.

Before clustering, we extended thus our vector space with concepts of a suitable ontology. In the Reuters scenario, we made use of WordNet¹², as its coverage fits to the generality of the Reuters news¹³. Then we applied nominal scaling and computed the concept lattice using the Cernato software of NaviCon GmbH¹⁴.

Figure 6 shows an example, where we scaled nominally with two thresholds: 7 % of the maximal value for (at least) *medium* importance, and 20 % of the maximal value for *high* importance. The diagram shows all clusters where the value of the concept (called synset in WordNet) ‘compound, chemical compound’ in the centroid is above the threshold of 7 %. Due to technical reasons, we reverse the usual reading order: A formal concept c_1 is a subconcept of a formal concept c_2 if and only if there is a path of descending(!) edges from the node representing c_1 to the node representing c_2 .

As an example, consider the concept in the lower middle of the diagram labeled by ‘oil’. It has {CL 3 (m), CL 9 (m), CL 23 (m), CL 79 (m), CL 85 (m), CL 95 (m)} as extent, and {organic compound, oil, ‘lipid, lipide, lipoid’, ‘compound, chemical compound’} as intent. Its intent indicates that the majority of documents within these clusters are about oil. This concept has three direct subconcepts: the first has {CL 3 (m)} as extent, and the attributes from above plus some attributes like ‘oil tanker’ and ‘Iranian’ as intent. The second has {CL 9 (m)} as extent, and the attributes from above plus some attributes like ‘area’, ‘palm’, and ‘metric ton’ as intent. The third subconcept has {CL 23 (m),

¹² <http://www.cogsci.princeton.edu/~wn/>

¹³ In [29], we discuss how different options for resolving ambiguities and for assigning more general concepts influence the quality of the clustering step.

¹⁴ <http://www.navicon.de>

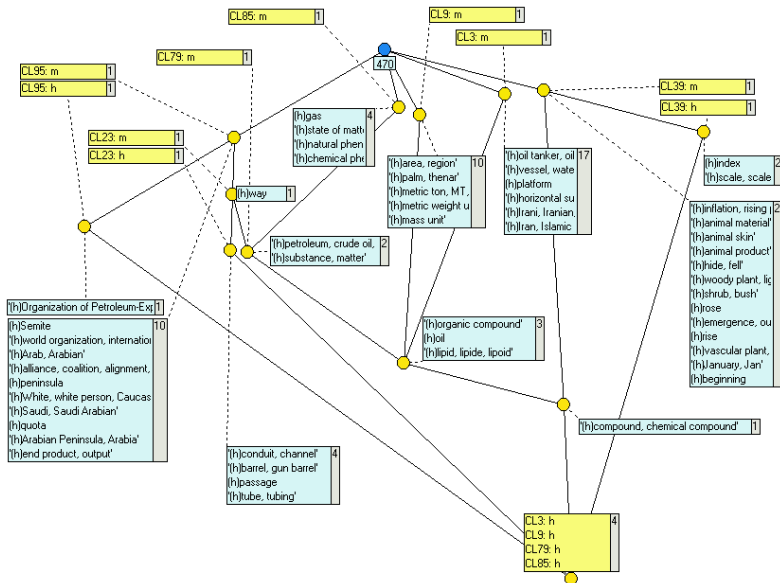


Fig. 6. The resulting conceptual clustering of the text clusters (visualized for the clusters related to chemical compounds)

CL 79 (m), CL 85 (m), CL 95 (m)} as extent, and the attributes from above plus ‘substance, matter’ as intent. These three subconcepts show what distinguishes the clusters related to oil. The majority of documents in Cluster 3 are about transport of oil (from Iran), those in Cluster 9 about (packaging of) palm oil, and those in the remaining clusters about crude oil.

Clustering the objects before applying FCA is an abstraction that might be considered a loss of information. However, it is beneficial for the following reasons. Firstly, it reduces the number of objects such that FCA becomes more efficient. Secondly, the technique is robust with regard to upcoming documents: A new document is first assigned to the cluster with the closest centroid, and then finds its place within the concept lattice. If on the contrary the document is considered directly for updating the concept lattice, there is no guarantee that the structure of the lattice does not change. Finally and most importantly, formal concept analysis applied directly on all documents suffers from the low co-occurrence of terms. The application of FCA on the Reuters-21578 dataset has shown that hardly any two texts generate the same formal concept. Thus, the lattice became large, unwieldy and hard to understand for the human user. Therefore, in our approach we first cluster a large number of texts (e.g. 10^5) into a more manageable number of clusters (e.g. 10^2). Only then we compute the lattice, allowing for abstraction from some randomness in the joint occurrences of terms. In this process, the use of an ontology as background knowledge assures consistency of the results.

5.3 Related Work

Logic has first been applied to extend and structure the set of attributes of a formal context in [55]. However, this approach considered all formulas of propositional logic

as attributes, which leads to a combinatorial explosion of the concept lattice. Chaudron/Maille [10] and Ferré/Ridoux [19] presented an approach based on first order logic. These approaches, however, consider all formulas of the given logic as attributes, which leads to an exponential explosion of the concept lattice. Using a T-Box allows to focus on the relevant formulas and to fine-tune the size of the resulting lattice. The way of computing the theory-driven logical scales is following the approach of Baader in [2]. In [48] is described how this computation can be extended from computing all conjunctions of attributes to computing both conjunctions and disjunctions.

In the future, FCA systems will be able to work on richer knowledge representations, including relational databases, DL, and conceptual graphs. First steps in this direction are already done with various extensions of the management system TOSCANA for Conceptual Information Systems [56]; consider, e. g., [54,53,36,50,25,1]. Within text mining, FCA techniques are also expected to be combined with successful techniques of that domain, such as latent semantic indexing [17] or concept indexing [33].

6 Conclusion

In this paper, we have shown by means of applications that Ontology Engineering and Formal Concept Analysis complement well; and that first steps have been made to set up links between the underlying theories. These links have to be strengthened and are to be exploited for establishing a comprehensive Conceptual Knowledge Processing environment. For future research within Formal Concept Analysis, this means that one has to extend the scope from strongly structured to semi-structured and even unstructured data, allowing to tackle more complex tasks as, e. g., in the Semantic Web. In this line it has become clear that when turning into the analysis of unstructured data and in particular textual data, a major challenge is to process the data in order to reduce their inherent complexity — e. g., by clustering — before actually applying FCA. By this we can reduce the processing time as well as yield more concise and human readable concept lattices also from unstructured data.

Furthermore, it has also become clear that the relation between formal concepts and ontology concepts is far from being trivial and intuitive. In some applications, FCA attributes are interpreted as ontology concepts (cf. Section 4.1), in others formal concepts are transformed into ontology concepts labeled by the intent of the former (cf. Section 3.2). As already mentioned, a major issue is thus to clarify this relation also from a theoretical point of view.

Having a closer look at the ontologies which were involved in our applications, we observe that most of them consist in fact only of a concept hierarchy. Relations have been used only in the Courseware Watchdog (Section 4.2) and in the logical scaling approach (Section 5.1); axioms have only been used in the latter. It is thus an interesting research question, in how far these more expressive knowledge representation features can be exploited further within Formal Concept Analysis.

Concluding, there is a high potential of interesting research questions about establishing and strengthening the link between the related research areas FCA and ontologies — both in theory and applications.

References

1. The ToscanaJ Project: An open-source reimplementation of Toscana.
<http://toscanaj.sourceforge.net>.
2. F. Baader. Computing a minimal representation of the subsumption lattice of all conjunctions of concept defined in a terminology. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Proc. Intl. KRUSE Symposium*, pages 168–178, Santa Cruz, August 11–13, 1995. UCSC.
3. R. Basili, M.T. Pazienza, and P. Velardi. Hierarchical clustering of verbs. In *Proceedings of the Workshop on Acquisition of Lexical Knowledge from Text*, 1993.
4. G. Bisson, C. Nedellec, and L. Canamero. Designing clustering methods for ontology building - The Mo'K workbench. In *Proceedings of the ECAI Ontology Learning Workshop*, 2000.
5. C. Brewster, F. Ciravegna, and Y. Wilks. Background and foreground knowledge in dynamic ontology construction. In *Proceedings of the SIGIR Semantic Web Workshop*, 2003.
6. S.A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguists*, pages 120–126, 1999.
7. C. Carpineto and G. Romano. Galois: An order-theoretic approach to conceptual clustering. In *Machine Learning. Proc. ICML 1993*, pages 33–40. Morgan Kaufmann Publishers, 1993.
8. H. Chalupsky. OntoMorph: A translation system for symbolic knowledge. In *Proc. 7th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 471–482, Breckenridge, Colorado, USA, April 2000.
9. E. Charniak and M. Berland. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 57–64, 1999.
10. Laurent Chaudron and Nicolas Maille. Generalized formal concept analysis. In B. Ganter and G. W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues. Proc. ICCS 2000*, volume LNAI 1867, pages 357–370, Heidelberg, 2000. Springer.
11. P. Cimiano, S. Staab, and J. Tane. Automatic acquisition of taxonomies from text: FCA meets NLP. In *Proceedings of the International Workshop on Adaptive Text Extraction and Mining*, 2003.
12. P. Cimiano, S. Staab, and J. Tane. Deriving concept hierarchies from text by smooth formal concept analysis. In *Proceedings of the GI Workshop "Lehren Lernen - Wissen - Adaptivität" (LLWA)*, 2003.
13. H. H. Clark. *Arenas of Language Use*, chapter Common Ground and Language Use, Definite Reference and Mutual Knowledge. CSLI, 1992.
14. R. Cole and P. Eklund. Browsing semi-structured web texts using formal concept analysis. In H. Delugach and G. Stumme, editors, *Conceptual Structures: Broadening the Base. Proc. ICCS '01*, pages 319–332, Heidelberg, 2001. Springer.
15. R. Cole, P. Eklund, and G. Stumme. Document retrieval for email search and discovery using formal concept analysis. *Journal of Applied Artificial Intelligence*, 17(3):257–280, 2003.
16. Richard Cole and Gerd Stumme. CEM - a Conceptual Email Manager. In Bernhard Ganter and Guy W. Mineau, editors, *Proc. ICCS 2000*, volume 1867 of LNAI, pages 438–452. Springer, 2000.
17. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
18. Errol Bozsak et al. KAON – Towards a Large Scale Semantic Web. In K. Bauknecht, A. Min Tjoa, and G. Quirchmayr, editors, *Proc. of the 3rd Intl. Conf. on E-Commerce and Web Technologies (EC-Web 2002)*, pages 304–313, 2002.
19. Sébastien Ferré and Olivier Ridoux. A logical generalization of formal concept analysis. In B. Ganter and G. W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues. Proc. ICCS 2000*, volume LNAI 1867, pages 371–384, Heidelberg, 2000. Springer.

20. Deutsches Institut für Normung. Begriffe und benennungen – allgemeine grundsätze. din 2330, 1993.
21. B. Ganter and G. Stumme. Creation and merging of ontology top-levels. In A. de Moor, W. Lex, and B. Ganter, editors, *Conceptual Structures for Knowledge Creation and Communication. Proc. ICCS '03*, volume LNAI 2746, pages 131–145, Heidelberg, 2003. Springer.
22. Petra Gast. Begriffliche Strukturen mehrwertiger Multikontexte. Master's thesis, FB Mathematik, TU Darmstadt, 1996.
23. T.R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *Formal Analysis in Conceptual Analysis and Knowledge Representation*. Kluwer, 1993.
24. M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, 1992.
25. J. Hereth and G. Stumme. Reverse pivoting in conceptual information systems.
26. W. Hesse. Ontologie(n) - Aktuelles Schlagwort. *Informatik Spektrum*, 25(6):477–480, 2002.
27. D. Hindle. Noun classification from predicate-argument structures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 268–275, 1990.
28. A. Hotho. *Clustern mit Hintergrundwissen*. PhD thesis, Institute AIFB, University of Karlsruhe, 2004. In preparation.
29. A. Hotho, S. Staab, and G. Stumme. Text clustering based on background knowledge. Technical report, University of Karlsruhe, Institute AIFB, 2003. 36 pages.
30. Andreas Hotho, Steffen Staab, and Gerd Stumme. Explaining text clustering results using semantic structures. In *Principles of Data Mining and Knowledge Discovery, 7th European Conference, PKDD 2003*, pages 217–228, Heidelberg, 2003. Springer.
31. E. Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proc. 1st Intl. Conf. on Language Resources and Evaluation*, Granada, Spain, May 1998.
32. K. Jones. View mail users manual. <http://www.wonderworks.com/vm>, 1999.
33. George Karypis and Eui-Hong Han. Fast supervised dimensionality reduction algorithm with applications to document categorization and retrieval. In *Proceedings of CIKM-00*, pages 12–19. ACM Press, New York, US, 2000.
34. R. E. Kent and C. Neuss. Creating a web analysis and visualization environment. *Computer Networks and ISDN Systems*, 28(1/2):109–117, 1995.
35. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proc. 7th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 483–493, Breckenridge, Colorado, USA, April 2000.
36. G. Mineau, G. Stumme, and R. Wille. Conceptual structures represented by conceptual graphs and formal concept analysis. In W. Tepfenhart and W. Cyre, editors, *Conceptual Structures: Standards and Practices. Proc. ICCS '99*, volume LNAI 1640, pages 423–441, Heidelberg, 1999. Springer.
37. N. Fridman Noy and M. A. Musen. PROMPT: algorithm and tool for automated ontology merging and alignment. In *Proc. 17th Natl. Conf. on Artificial Intelligence (AAAI'2000)*, pages 450–455, Austin, TX, July/August 2000.
38. F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, 1993.
39. Wiebke Petersen. A set-theoretical approach for the induction of inheritance hierarchies. *Electronic Notes in Theoretical Computer Science*, 51, 2002.
40. S. Prediger. Logical scaling in formal concept analysis. In D. Lukose, H. Delugach, M. Keeler, L. Searle, and J.F. Sowa, editors, *Conceptual structures: Fulfilling Peirce's dream*, Heidelberg. Springer.
41. S. Prediger and G. Stumme. Theory-driven logical scaling. conceptual information systems meet description logics. In E. Franconi et al, editor, *Proc. 6th Intl. Workshop Knowledge Representation Meets Databases*, Heidelberg. CEUR Workshop Proc.

42. S. Prediger and R. Wille. The lattice of concept graphs of a relationally scaled context. In W. Tepfenhart and W. Cyre, editors, *Conceptual Structures: Standards and Practices. Proc. ICCS '99*, volume LNAI 1640, pages 401–414, Heidelberg, 1999. Springer.
43. Uta Priss. Linguistic applications of formal concept analysis. presentation at the First International Conference on Formal Concept Analysis, Darmstadt, 2003.
44. D. Richards and P. Compton. Combining formal concept analysis and ripple down rules to support reuse. In *Proc. 9th Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE '97)*, Heidelberg, 1997. Springer.
45. Christoph Schmitz, Steffen Staab, Rudi Studer, Gerd Stumme, and Julien Tane. Accessing distributed learning repositories through a courseware watchdog. In M. Driscoll and T. C. Reeves, editors, *Proc. of E-Learn 2002 World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education (E-Learn 2002)*, pages 909–915, Norfolk, 2002. AACE.
46. S. Schulte im Walde. Clustering verbs semantically according to their alternation behaviour. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, 2000.
47. Caroline Sporleder. A galois lattice based approach to lexical inheritance hierarchy learning. In *Proceedings of the ECAI 2002 Workshop on Machine Learning and Natural Language Processing for Ontology Engineering (OLT 2002)*, 2002.
48. G. Stumme. The concept classification of a terminology extended by conjunction and disjunction. In N. Foo and R. Goebel, editors, *PRICAI '96: Topics in Artificial Intelligence*, Heidelberg. Springer.
49. G. Stumme. Local scaling in conceptual data systems. In P. W. Eklund, G. Ellis, and G. Mann, editors, *Conceptual Structures: Knowledge Representation as Interlingua. Proc. ICCS 96*, volume LNAI 1115, pages 308–320, Heidelberg, 1996. Springer.
50. G. Stumme. Chapter 14: Conceptual On-Line Analytical Processing. In K. Tanaka, S. Ghandeharizadeh, and Y. Kambayashi, editors, *Information Organization and Databases*, pages 191–203, Boston–Dordrecht–London, 2000. Kluwer.
51. G. Stumme and A. Maedche. FCA–Merge: Bottom-up merging of ontologies. In *Proc. 17th Intl. Conf. on Artificial Intelligence (IJCAI '01)*, pages 225–230, Seattle, WA, USA, 2001.
52. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing iceberg concept lattices with Titanic. *J. on Knowledge and Data Engineering*, 42(2):189–222, 2002.
53. G. Stumme and K. E. Wolff. Numerical aspects in the data model of conceptual information systems. In Y. Kambayashi, Dik Kun Lee, Ee-Peng Lim, M. K. Mohania, and Y. Masunaga, editors, *Advances in Database Technologies. Proc. Intl. Workshop on Data Warehousing and Data Mining, 17th Intl. Conf. on Conceptual Modeling (ER '98)*, volume LNCS1552, pages 117–228, Heidelberg, 1999. Springer.
54. G. Stumme and K. E. Wolff. Computing in conceptual data systems with relational structures. In *Proc. Intl. Conf. on Knowledge Retrieval, Use, and Storage for Efficiency*, pages 206–219, Vancouver, Canada, August 11–13, 1997.
55. Gerd Stumme. Boolesche begriffe. Master's thesis, FB Mathematik, TU Darmstadt, 1994.
56. F. Vogt and R. Wille. TOSCANA – a graphical tool for analyzing and exploring data. In R. Tamassia and I. G. Tollis, editors, *GraphDrawing '94*, volume LNCS 894, pages 226–233, Heidelberg, 1995. Springer.
57. R. Wille. Preconcept algebras and generalized double boolean algebras. In *This volume*.
58. R. Wille. Conceptual structures of multicontexts. In P. W. Eklund, G. Ellis, and G. Mann, editors, *Conceptual Structures: Knowledge Representation as Interlingua. Proc. ICCS '96*, volume LNAI 1115, pages 23–39, Heidelberg, 1996. Springer.
59. R. Wille. Conceptual graphs and formal concept analysis. In D. Lukose, H. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *Conceptual Structures: Fulfilling Peirce's Dream. Proc. ICCS '97*, volume LNAI 1257, pages 290–303, Heidelberg, 1997. Springer.
60. G. Zipf. *Selective Studies and the Principle of Relative Frequency in Language*. Cambridge, 1932.

A First Step towards Protoconcept Exploration

Björn Vormbrock

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D-64289 Darmstadt
vormbrock@mathematik.tu-darmstadt.de

Abstract. Protoconcept exploration is developed as a knowledge acquisition tool for exploring the structure of protoconcept algebras similar to concept exploration for concept lattices. In this paper first results are presented. In particular, an algorithm is introduced that interactively determines the finitely generated subalgebra of a protoconcept algebra. Although it creates a redundant set of questions it shows clearly which information from the user is needed and it serves as a basis for a future, optimized algorithm.

1 Introduction

The definition and investigation of protoconcept algebras arose from the development of Boolean Concept Logic as one part of Contextual Logic. The aim of Contextual Logic is the mathematization of the classical philosophical logic with its doctrines of concepts, judgments, and conclusions. Boolean Concept Logic is an approach to mathematize the logic of concepts based on Formal Concept Analysis. We assume that the reader is familiar with the basic notions of Formal Concept Analysis. An extensive introduction to Formal Concept Analysis is [GW99], an introduction to Contextual Logic can be found in [Wi00b]. Recent, more specialized works on Boolean Concept Logic are [HLSW00], [Wi00a], [Vo03], [KV03] and [VW03]. Protoconcepts were introduced in [Wi00a] as generalizations of formal concepts in order to permit negations on concepts:

Definition 1. A protoconcept of a formal context $\mathbb{K} := (G, M, I)$ is a pair (A, B) with $A \subseteq G$ and $B \subseteq M$ such that $A' = B''$ or, equivalently, $A'' = B'$. We denote the set of all protoconcepts of a context \mathbb{K} by $\mathfrak{P}(\mathbb{K})$ and define on $\mathfrak{P}(\mathbb{K})$ operations $\sqcap, \sqcup, \neg, \neg, \top$ and \perp by:

$$\begin{aligned}(A_1, B_1) \sqcap (A_2, B_2) &:= (A_1 \cap A_2, (A_1 \cap A_2)') \\ (A_1, B_1) \sqcup (A_2, B_2) &:= ((B_1 \cap B_2)', B_1 \cap B_2) \\ \neg(A, B) &:= (G \setminus A, (G \setminus A)') \\ \neg(A, B) &:= ((M \setminus B)', M \setminus B) \\ \top &:= (G, \emptyset) \\ \perp &:= (\emptyset, M)\end{aligned}$$

The set of all protoconcepts of a context \mathbb{K} together with these operations is called the protoconcept algebra of \mathbb{K} and denoted by $\underline{\mathfrak{P}}(\mathbb{K})$.

The operations are called “meet” (\sqcap), “join” (\sqcup), “negation” (\neg), “opposition” (\neg), “all” (\top) and “nothing” (\perp). A small example of a protoconcept algebra is depicted in Fig. 1. Each protoconcept (A, B) determines a unique concept of the context namely $(A'', A') (= (B', B''))$. Therefore, we can understand the extent A as a set of prototypic objects and the intent B as a set of characteristic attributes determining this concept (cf. [VW03]). The introduction of a negation yields a richer structure and thus a greater expressiveness of protoconcepts. An example of practical interest that illustrates this expressiveness and its advantages can be found in [KV03]. Since not only concepts, but also “opposites” and negations of concepts form part of our thinking, it is natural to ask how they can be used for the exploration of a knowledge landscape.

Protoconcept exploration is developed similarly to concept exploration as described in [St97]. It deals with the following problem: While we can easily compute the protoconcepts of a given context, there may be situations where one has only implicit knowledge of a domain of interest and it is nearly impossible to write down explicitly all objects, all attributes and their incidence relation in a context. As an example from mathematics consider the set $\mathbb{R}^{\mathbb{R}}$ of all functions from \mathbb{R} to \mathbb{R} and a set of mathematically defined properties these functions can have. It is impossible to list all the objects and attributes, however some of its protoconcepts can be considered as ‘fairly clear’, like for example the protoconcepts (m', m) generated by attributes m such as ‘continuous’, ‘differentiable’, ‘polynomial function’, and protoconcepts (g, g') generated by objects g such as *identity*, the constant function $x \mapsto 3$ and the exponential function $x \mapsto e^x$. Protoconcept exploration is a knowledge acquisition tool that helps an investigator to compute the protoconcepts of the underlying protoconcept algebra that can be obtained from these “fairly clear” basic protoconcepts. Thus, the aim of protoconcept exploration can be stated as follows (cp. [St97]): We assume that there exists an underlying protoconcept algebra $\underline{\mathfrak{P}}_u = \mathfrak{P}(G_u, M_u, I_u)$ which may be only ‘vaguely known’ by the investigator. For a finite subset $\mathcal{B} = \{p_1, \dots, p_n\}$ of $\underline{\mathfrak{P}}_u$, the set of basic protoconcepts, protoconcept exploration shall support him in determining the structure of the subalgebra $\underline{\mathfrak{P}}_g$ of $\underline{\mathfrak{P}}_u$ which is generated by \mathcal{B} . Therefore, the algorithm generates questions about implications between the identified protoconcepts. The answers given by the user are then used to find more protoconcepts. It is not necessary that the investigator has explicit knowledge of the extent or the intent of the basic protoconcepts, but she has to decide whether basic protoconcepts and protoconcepts computed during the generation process have objects or attributes in common or not.

The main differences between concept exploration as in [St97] and protoconcept exploration are due to the much richer structure of protoconcept algebras. The additional operations negation and opposition allow the generation of more protoconcepts, including more concepts, than the operations meet and join. This implies that even a single protoconcept can generate an infinite protoconcept algebra while one needs at least three concepts to generate an infinite concept lattice. In effect, the algorithm for protoconcept exploration will not necessarily terminate if the underlying context is infinite.

We consider this paper a first step towards an algorithm for protoconcept exploration because we have focused on developing an algorithm that is mathematically correct and shows clearly which information is needed from the user. Yet it is far from being user-friendly since it produces a lot of redundant questions. Nevertheless we are confident that the presented algorithm is the basis for an algorithm that can be implemented and used. Some ideas for improvement are described in the section “Further Research”.

In section 2, we first present definitions and basic properties of protoconcept algebras. In the third section implications between protoconcepts and sequents of protoconcepts are introduced. This provides us with the notions to formulate the algorithm for protoconcept exploration and to prove its correctness in section 4.

2 Protoconcept Algebras and Double Boolean Algebras

For the understanding of protoconcept algebras it is useful to define additional operations on protoconcepts:

$$\begin{aligned}(A_1, B_1) \sqcup (A_2, B_2) &:= \neg(\neg(A_1, B_1) \sqcap \neg(A_2, B_2)) \text{ and} \\ (A_1, B_1) \sqcap (A_2, B_2) &:= \neg(\neg(A_1, B_1) \sqcup \neg(A_2, B_2)), \\ \top &:= \neg \perp \text{ and } \perp := \neg \top.\end{aligned}$$

Note that the result of any operation on protoconcepts is always of the form (A, A') or (B', B) . Protoconcepts of the form (A, A') are called \sqcap -semiconcepts, those of type (B', B) are called \sqcup -semiconcepts. The set of all \sqcap -semiconcepts of \mathbb{K} is denoted by $\mathfrak{H}_{\sqcap}(\mathbb{K})$ and, likewise, the set of all \sqcup -semiconcepts of \mathbb{K} is denoted by $\mathfrak{H}_{\sqcup}(\mathbb{K})$. Obviously, the set of all semiconcepts $\mathfrak{H}(\mathbb{K}) := \mathfrak{H}_{\sqcap}(\mathbb{K}) \cup \mathfrak{H}_{\sqcup}(\mathbb{K})$ together with the operations of $\mathfrak{P}(\mathbb{K})$ forms a subalgebra of $\mathfrak{P}(\mathbb{K})$ called the *semiconcept algebra* $\underline{\mathfrak{H}}(\mathbb{K})$ of \mathbb{K} . The set $\mathfrak{H}_{\sqcap}(\mathbb{K})$ is closed under the operations \sqcap , \sqcup , \neg , \perp and \top ; therefore $\underline{\mathfrak{H}}_{\sqcap}(\mathbb{K}) := (\mathfrak{H}_{\sqcap}(\mathbb{K}), \sqcap, \sqcup, \neg, \perp, \top)$ is a Boolean algebra isomorphic to the powerset algebra of G . Dually, $\underline{\mathfrak{H}}_{\sqcup}(\mathbb{K}) := (\mathfrak{H}_{\sqcup}(\mathbb{K}), \sqcup, \sqcap, \neg, \perp, \top)$ is a Boolean algebra antiisomorphic to the powerset algebra of M . Moreover, (A, B) is a concept of \mathbb{K} if and only if it is both a \sqcup - and a \sqcap -semiconcept. Figure 1 depicts a context and its protoconcept algebra. The elements represented by filled circles are formal concepts. The circles with the upper half filled represent \sqcup -semiconcepts, those with the lower half filled represent \sqcap -semiconcepts.

Theorem 1 ([Wi00a]). *A basis for all equations which are valid in all protoconcept algebras is given by the following equations which are the equational axioms of the so-called double Boolean algebras:*

1a) $(x \sqcap x) \sqcap y = x \sqcap y$	1b) $(x \sqcup x) \sqcup y = x \sqcup y$
2a) $x \sqcap y = y \sqcap x$	2b) $x \sqcup y = y \sqcup x$
3a) $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$	3b) $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$
4a) $x \sqcap (x \sqcup y) = x \sqcap x$	4b) $x \sqcup (x \sqcap y) = x \sqcup x$
5a) $x \sqcap (x \sqcup y) = x \sqcap x$	5b) $x \sqcup (x \sqcap y) = x \sqcup x$
6a) $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$	6b) $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$
7a) $\neg \neg(x \sqcap y) = x \sqcap y$	7b) $\neg \neg(x \sqcup y) = x \sqcup y$

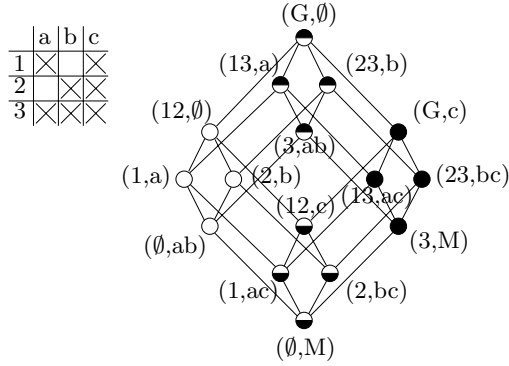


Fig. 1. A context and its protoconcept algebra

$$8a) \neg(x \sqcap x) = \neg x$$

$$9a) x \sqcap \neg x = \perp$$

$$10a) \neg \perp = \top \sqcap \top$$

$$11a) \neg \top = \perp$$

$$12) (x \sqcap x) \sqcup (x \sqcap x) = (x \sqcup x) \sqcap (x \sqcup x)$$

$$8b) \neg(x \sqcup x) = \neg x$$

$$9b) x \sqcup \neg x = \top$$

$$10b) \neg \top = \perp \sqcup \perp$$

$$11b) \neg \perp = \top$$

with the operations $\sqcup, \sqcap, \neg, \perp$ defined as $x \sqcup y := \neg(\neg x \sqcap \neg y)$, $x \sqcap y := \neg(\neg x \sqcup \neg y)$, $\neg := \neg \perp$ and $\perp := \neg \top$.

An algebra $\underline{D} := (D, \sqcap, \sqcup, \neg, \neg, \perp, \top)$ of type $(2,2,1,1,0,0)$ satisfying the equations 1a) to 12) is called a *double Boolean algebra*. Obviously, every protoconcept algebra is a double Boolean algebra. Semiconcept algebras satisfy also

$$13) x = x \sqcap x \text{ or } x = x \sqcup x$$

and double Boolean algebras that satisfy 13) are called *pure double Boolean algebras*.

We define a quasi-order \sqsubseteq on double Boolean algebras by $x \sqsubseteq y \Leftrightarrow x \sqcap y = x \sqcap x$ and $x \sqcup y = y \sqcup y$. For protoconcepts \sqsubseteq is an order and equivalent to $(A_1, B_1) \sqsubseteq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ and $B_2 \subseteq B_1$. For any protoconcept (A, B) , $(A, B) \sqcap (A, B) = (A, A')$ yields the greatest \sqcap -semiconcept below (A, B) and, dually, $(A, B) \sqcup (A, B) = (B', B)$ yields the smallest \sqcup -semiconcept greater than (A, B) . To shorten notation, we set $x_{\sqcap} := x \sqcap x$ and $x_{\sqcup} := x \sqcup x$ for an element x of a double Boolean algebra \underline{D} , and write $D_{\sqcap} := \{x_{\sqcap} \mid x \in D\}$ and, dually, $D_{\sqcup} := \{x_{\sqcup} \mid x \in D\}$. As in the case of protoconcept algebras, $\underline{D}_{\sqcap} := (D_{\sqcap}, \sqcap, \sqcup, \neg, \neg, \perp, \top)$ and $\underline{D}_{\sqcup} := (D_{\sqcup}, \sqcup, \sqcap, \neg, \neg, \perp, \top)$ form Boolean algebras. A double Boolean algebra \underline{D} is called *complete* if and only if the Boolean algebras \underline{D}_{\sqcap} and \underline{D}_{\sqcup} are complete. In [VW03], basic theorems on semiconcept algebras and on protoconcept algebras are proven, characterizing those double Boolean algebras which can be considered semiconcept resp. protoconcept algebras of an appropriate context. For this paper we need the Basic Theorem on Semiconcept Algebras:

Theorem 2 (The Basic Theorem on Semiconcept Algebras, [VW03]). For a context $\mathbb{K} := (G, M, I)$, the semiconcept algebra $\underline{\mathfrak{H}}(\mathbb{K})$ is a complete pure double Boolean algebra whose Boolean algebras $\underline{\mathfrak{H}}_{\sqcap}(\mathbb{K})$ and $\underline{\mathfrak{H}}_{\sqcup}(\mathbb{K})$ are atomic. The (arbitrary) meet and join of $\underline{\mathfrak{H}}(\mathbb{K})$ are given by

$$\bigcap_{t \in T} (A_t, B_t) = \left(\bigcap_{t \in T} A_t, \left(\bigcap_{t \in T} A_t \right)' \right) \quad \text{and} \quad \bigsqcup_{t \in T} (A_t, B_t) = \left(\left(\bigcap_{t \in T} B_t \right)', \bigcap_{t \in T} B_t \right).$$

In general, a complete pure double Boolean algebra \underline{D} whose Boolean algebras \underline{D}_{\sqcap} and \underline{D}_{\sqcup} are atomic, is isomorphic to $\underline{\mathfrak{H}}(\mathbb{K})$ if and only if there exist a bijection $\tilde{\gamma}$ from G onto the set $\mathcal{A}(\underline{D}_{\sqcap})$ of all atoms of \underline{D}_{\sqcap} and a bijection $\tilde{\mu}$ from M onto the set $\mathcal{C}(\underline{D}_{\sqcup})$ of all coatoms of \underline{D}_{\sqcup} such that $gIm \Leftrightarrow \tilde{\gamma}(g) \sqsubseteq \tilde{\mu}(m)$ for all $g \in G$ and $m \in M$. In particular, for any complete pure double Boolean algebra \underline{D} whose Boolean algebras are atomic, we get $\underline{D} \cong \underline{\mathfrak{H}}(\mathcal{A}(\underline{D}_{\sqcap}), \mathcal{C}(\underline{D}_{\sqcup}), \sqsubseteq)$, i.e., the semiconcept algebras are up to isomorphism the complete pure double Boolean algebras \underline{D} whose Boolean algebras \underline{D}_{\sqcap} and \underline{D}_{\sqcup} are atomic.

Now the subalgebra of a protoconcept algebra that is generated by a set of protoconcepts $\{\mathfrak{p}_1, \dots, \mathfrak{p}_n\}$ is the union of the pure subalgebra of its semiconcept algebra generated by $\{\mathfrak{p}_{1\sqcap}, \mathfrak{p}_{1\sqcup}, \dots, \mathfrak{p}_{n\sqcap}, \mathfrak{p}_{n\sqcup}\}$ and the set $\{\mathfrak{p}_1, \dots, \mathfrak{p}_n\}$. If this subalgebra $\underline{\mathfrak{P}}_g^p$ of the semiconcept algebra is finite, then the Basic Theorem on Semiconcept Algebras yields that $\underline{\mathfrak{P}}_g^p$ is the semiconcept algebra $\underline{\mathfrak{H}}(\mathcal{A}(\underline{\mathfrak{P}}_{g\sqcap}^p), \mathcal{C}(\underline{\mathfrak{P}}_{g\sqcup}^p), \sqsubseteq)$. Therefore, our strategy will be to compute $\mathcal{A}(\underline{\mathfrak{P}}_{g\sqcap}^p)$ and $\mathcal{C}(\underline{\mathfrak{P}}_{g\sqcup}^p)$. We investigate the implications between protoconcepts in order to find these sets.

3 Implications and Sequents

Definition 2. We say that a protoconcept (A_1, B_1) implies a protoconcept (A_2, B_2) $((A_1, B_1) \rightarrow (A_2, B_2))$ iff $(A_1, B_1) \sqsubseteq (A_2, B_2)$ $(\Leftrightarrow A_1 \subseteq A_2 \text{ and } B_2 \subseteq B_1)$.

We are especially interested in implications $\bigcap \mathfrak{A} \rightarrow \bigcup \mathfrak{B}$ for $\mathfrak{A}, \mathfrak{B} \subseteq \underline{\mathfrak{H}}_{\sqcup}(\mathbb{K})$ and implications $\bigcap \mathfrak{A} \rightarrow \bigcup \mathfrak{B}$ for $\mathfrak{A}, \mathfrak{B} \subseteq \underline{\mathfrak{H}}_{\sqcap}(\mathbb{K})$.

Definition 3. For a protoconcept algebra $\underline{\mathfrak{P}}(\mathbb{K})$ and sets $S \subseteq \underline{\mathfrak{H}}_{\sqcup}(\mathbb{K})$ and $T \subseteq \underline{\mathfrak{H}}_{\sqcap}(\mathbb{K})$ we say that:

1. The pair $(\mathfrak{A}, \mathfrak{B}) \subseteq S \times S$ is a \sqcup -sequent over S if $\mathfrak{A} \cap \mathfrak{B} = \emptyset$.
2. Dually, the pair $(\mathfrak{A}, \mathfrak{B}) \subseteq T \times T$ is a \sqcap -sequent over T if $\mathfrak{A} \cap \mathfrak{B} = \emptyset$,
3. A \sqcup -sequent over S (\sqcap -sequent over T) is full iff $\mathfrak{A} \cup \mathfrak{B} = S$ ($\mathfrak{A} \cup \mathfrak{B} = T$).

On sequents of the same type we define an order by $(\mathfrak{A}_1, \mathfrak{B}_1) \leq (\mathfrak{A}_2, \mathfrak{B}_2) : \Leftrightarrow \mathfrak{A}_1 \subseteq \mathfrak{A}_2$ and $\mathfrak{B}_1 \subseteq \mathfrak{B}_2$. Obviously, if for \sqcup -sequents (resp. \sqcap -sequents) $(\mathfrak{A}_1, \mathfrak{B}_1) \leq (\mathfrak{A}_2, \mathfrak{B}_2)$ and $\bigcap \mathfrak{A}_1 \rightarrow \bigcup \mathfrak{B}_1$ (resp. $\bigcap \mathfrak{A}_1 \rightarrow \bigcup \mathfrak{B}_1$) in a protoconcept algebra then also $\bigcap \mathfrak{A}_2 \rightarrow \bigcup \mathfrak{B}_2$ (resp. $\bigcap \mathfrak{A}_2 \rightarrow \bigcup \mathfrak{B}_2$). We understand \sqcup -sequents

$(\mathfrak{A}, \mathfrak{B})$ as implications $\sqcap \mathfrak{A} \rightarrow \sqcup \mathfrak{B}$ and \sqcap -sequents $(\mathfrak{A}, \mathfrak{B})$ as implications $\sqcap \mathfrak{A} \rightarrow \sqcup \mathfrak{B}$. Note that:

$$\sqcap \mathfrak{A} \rightarrow \sqcup \mathfrak{B} \Leftrightarrow \neg \sqcap \mathfrak{A} \sqcup \sqcup \mathfrak{B} = \top \quad (1)$$

$$\sqcap \mathfrak{A} \rightarrow \sqcup \mathfrak{B} \Leftrightarrow \sqcap \mathfrak{A} \sqcap \neg \sqcup \mathfrak{B} = \perp. \quad (2)$$

In order to find $\mathcal{C}(\mathfrak{P}_{g\sqcup}^p)$, we explore the logic of \mathfrak{P}_g^p and learn from invalid implications $\sqcap \mathfrak{A} \rightarrow \sqcup \mathfrak{B}$ that $\neg \sqcap \mathfrak{A} \sqcup \sqcup \mathfrak{B} \neq \top$. Hence, we find that there is a “counterexample” to the implication a \sqcup -semiconcept $x = \neg \sqcap \mathfrak{A} \sqcup \sqcup \mathfrak{B} \sqsubset \top$. Dually, we use the invalid implications $\sqcap \mathfrak{A} \rightarrow \sqcup \mathfrak{B}$ and equation 2 to find small \sqcap -semiconcepts above \perp . In order to make the formulas better readable we set for \sqcup -sequents $(\mathfrak{A}, \mathfrak{B})$: $j(\mathfrak{A}, \mathfrak{B}) := \neg \sqcap \mathfrak{A} \sqcup \sqcup \mathfrak{B}$ and for \sqcap -sequents $(\mathfrak{A}, \mathfrak{B})$: $m(\mathfrak{A}, \mathfrak{B}) := \sqcap \mathfrak{A} \sqcap \neg \sqcup \mathfrak{B}$.

For a full sequent $(\mathfrak{A}, \mathfrak{B})$ we find a close relation between $m(\mathfrak{A}, \mathfrak{B})$ (resp. $j(\mathfrak{A}, \mathfrak{B})$) and partitions of the object (resp. attribute) set of the underlying context. First we recall the definition of partitions as given in [Gr68].

Definition 4. A partition π of a set A is a subset of $\mathfrak{P}(A)$ satisfying: Every $a \in A$ is an element of exactly one $B \in \pi$. The members of π are called blocks of the partition

On the set $Part(A)$ of all partitions of a set A we can define an order relation \leq by: $\pi_0 \leq \pi_1$ if for every block B of π_0 , there exists a block C of π_1 , with $B \subseteq C$. In this case π_0 is a *refinement* of π_1 . The order relation \leq on $Part(A)$ makes it a complete lattice.

Proposition 1. For a set S of \sqcap -semiconcepts of a context $\mathbb{K} := (G, M, I)$, the extents of the semiconcepts $m(\mathfrak{A}, \mathfrak{B}) \neq \perp$ for full \sqcap -sequents $(\mathfrak{A}, \mathfrak{B})$ over S yield a partition π_S of G . Moreover, in the lattice $Part(G)$, the partition π_S is the infimum of all partitions $\pi_A := \{A, G \setminus A\}$ for $(A, A') \in S$.

Dually, for a set T of \sqcup -semiconcepts of \mathbb{K} , the intents of the semiconcepts $j(\mathfrak{A}, \mathfrak{B}) \neq \top$ for full \sqcup -sequents $(\mathfrak{A}, \mathfrak{B})$ over T yield a partition π_T of M . In the lattice $Part(M)$, the partition π_T is the infimum of all partitions $\pi_B := \{B, M \setminus B\}$ for $(B', B) \in T$.

Proof: Obviously, the π_A are partitions of G . The blocks of the infimum $\pi_i := \bigwedge \{\pi_A \mid (A, A') \in S\}$ are the nonempty sets obtained as $b_{S'} := \bigcap_{(A, A') \in S'} A \cap \bigcap_{(A, A') \in (S \setminus S')} G \setminus A$ from all subsets $S' \subseteq S$. Since

$$\begin{aligned} m(\mathfrak{A}, \mathfrak{B}) &= \sqcap \mathfrak{A} \sqcap \neg \sqcup \mathfrak{B} \\ &= ((\bigcap_{(A, A') \in \mathfrak{A}} A) \cap (G \setminus \bigcup_{(A, A') \in \mathfrak{B}} A), ((\bigcap_{(A, A') \in \mathfrak{A}} A) \cap (G \setminus \bigcup_{(A, A') \in \mathfrak{B}} A))') \\ &= ((\bigcap_{(A, A') \in \mathfrak{A}} A) \cap (\bigcap_{(A, A') \in \mathfrak{B}} G \setminus A), ((\bigcap_{(A, A') \in \mathfrak{A}} A) \cap (\bigcap_{(A, A') \in \mathfrak{B}} G \setminus A))'), \end{aligned}$$

the block $b_{S'}$ corresponds to the extent of $m(S', S \setminus S')$ and $b_{S'} \neq \emptyset \Leftrightarrow m(S', S \setminus S') \neq \perp$. Thus for every full \sqcap -sequent $(\mathfrak{A}, \mathfrak{B})$ over S satisfying $m(\mathfrak{A}, \mathfrak{B}) \neq \perp$, the extent of $m(\mathfrak{A}, \mathfrak{B})$ is a block of π_i and conversely, for any block $b_{S'}$ of π_i , $(S', S \setminus S')$ is a full \sqcap -sequent over S such that $b_{S'}$ is the extent of $m(S', S \setminus S')$. Analogously we obtain the correspondence between the full \sqcup -sequents over T and the blocks of the partition π_T of M . \square

It is evident that, for sets S, T of semiconcepts, $S \subseteq T$ implies $\pi_T \leq \pi_S$. For a family $(\mathfrak{A}_i, \mathfrak{B}_i)_{i \in I}$ of full \sqcap -sequents over a set S , the extent of $\bigsqcup_{i \in I} m(\mathfrak{A}_i, \mathfrak{B}_i)$ is the union of the blocks of π_S corresponding to the extents of the $m(\mathfrak{A}_i, \mathfrak{B}_i)$. Since for a partition the intersection of two unions of blocks is the union of their common blocks, we obtain immediately:

Corollary 1. 1) Let S be a set of \sqcap -semiconcept and let X, Y be two sets of full \sqcap -sequents over S satisfying: $(\mathfrak{A}, \mathfrak{B}) \in X \cup Y \Rightarrow m(\mathfrak{A}, \mathfrak{B}) \neq \perp$. Then $\bigsqcup m(X) \sqcap \bigsqcup m(Y) = \bigsqcup m(X \cap Y)$.

2) Dually, for a set T of \sqcup -semiconcepts and sets X, Y of full \sqcup -sequents over S satisfying $(\mathfrak{A}, \mathfrak{B}) \in X \cup Y \Rightarrow j(\mathfrak{A}, \mathfrak{B}) \neq \top$ we obtain $\bigsqcap j(X) \sqcup \bigsqcap j(Y) = \bigsqcap j(X \cap Y)$

For a \sqcup -sequent (resp. \sqcap -sequent) $(\mathfrak{A}, \mathfrak{B})$ over S , we can express the intent (extent) of $j(\mathfrak{A}, \mathfrak{B})$ ($m(\mathfrak{A}, \mathfrak{B})$) as the union of blocks of the partition π_S :

Proposition 2. 1) Let $(\mathfrak{A}, \mathfrak{B})$ be a \sqcup -sequent over a set S of \sqcup -semiconcepts. Then

$$j(\mathfrak{A}, \mathfrak{B}) = \bigsqcap \{j(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \text{ is a full } \sqcup\text{-sequent over } S \text{ with } j(\mathfrak{C}, \mathfrak{D}) \neq \top \text{ and } (\mathfrak{A}, \mathfrak{B}) \leq (\mathfrak{C}, \mathfrak{D})\}.$$

2) Dually, let $(\mathfrak{A}, \mathfrak{B})$ be a \sqcap -sequent over a set T of \sqcap -semiconcepts. Then

$$m(\mathfrak{A}, \mathfrak{B}) = \bigsqcup \{m(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \text{ is a full } \sqcap\text{-sequent over } T \text{ with } m(\mathfrak{C}, \mathfrak{D}) \neq \perp \text{ and } (\mathfrak{A}, \mathfrak{B}) \leq (\mathfrak{C}, \mathfrak{D})\}.$$

Proof: First, we show that from $(\mathfrak{A}, \mathfrak{B}) \leq (\mathfrak{C}, \mathfrak{D})$ follows $j(\mathfrak{A}, \mathfrak{B}) \sqsubseteq j(\mathfrak{C}, \mathfrak{D})$: $j(\mathfrak{C}, \mathfrak{D}) = \neg \bigsqcap \mathfrak{C} \sqcup \bigsqcup \mathfrak{D} = \neg \bigsqcap (\mathfrak{C} \setminus \mathfrak{A}) \sqcup \neg \bigsqcap \mathfrak{A} \sqcup \bigsqcup (\mathfrak{D} \setminus \mathfrak{B}) \sqcup \bigsqcup \mathfrak{B} = j(\mathfrak{A}, \mathfrak{B}) \sqcup \neg \bigsqcap (\mathfrak{C} \setminus \mathfrak{A}) \sqcup \bigsqcup (\mathfrak{D} \setminus \mathfrak{B}) \supseteq j(\mathfrak{A}, \mathfrak{B})$. This yields

$$j(\mathfrak{A}, \mathfrak{B}) \sqsubseteq \bigsqcap \{j(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \text{ is a full } \sqcup\text{-sequent over } S \text{ with } j(\mathfrak{C}, \mathfrak{D}) \neq \top \text{ and } (\mathfrak{A}, \mathfrak{B}) \leq (\mathfrak{C}, \mathfrak{D})\} =: (B'_{inf}, B_{inf}).$$

Now let $(B', B) = j(\mathfrak{A}, \mathfrak{B})$. For an attribute $m \in B$ and a \sqcup -semiconcept (D', D) in S we have that $(D', D) \in \mathfrak{B} \Rightarrow m \in D$ and that $(D', D) \in \mathfrak{A} \Rightarrow m \notin D$. We set $\mathfrak{C}_m := \{(D', D) \in S \mid m \notin D\}$ and $\mathfrak{D}_m := \{(D', D) \in S \mid m \in D\}$. Then $(\mathfrak{C}_m, \mathfrak{D}_m)$ is a full \sqcup -sequent over S satisfying $(\mathfrak{C}_m, \mathfrak{D}_m) \geq (\mathfrak{A}, \mathfrak{B})$. Thus m is an element of the intent of $j(\mathfrak{C}_m, \mathfrak{D}_m)$ and therefore $m \in B_{inf}$, which yields $j(\mathfrak{A}, \mathfrak{B}) = (B'_{inf}, B_{inf})$. The second part follows likewise. \square

4 An Algorithm for Protoconcept Exploration

Now we have the means to formulate the algorithm. Instead of computing all the generated semiconcepts, we use that all elements of $\mathfrak{P}_{g\sqcup}^p$ can be obtained from the coatoms $\mathcal{C}(\underline{\mathfrak{P}}_{g\sqcup}^p)$ and, dually, all elements of $\mathfrak{P}_{g\sqcap}^p$ can be obtained from the atoms $\mathcal{A}(\underline{\mathfrak{P}}_{g\sqcap}^p)$. In every iteration we compute greater elements of $\mathfrak{P}_{g\sqcup}^p$ (the sets M_i) and smaller elements of $\mathfrak{P}_{g\sqcap}^p$ (the sets G_i) using the smallest (resp. greatest) elements of the previous step \widetilde{G}_{i-1} (resp. \widetilde{M}_{i-1}). The algorithm stops after an iteration if no new protoconcepts were found.

The algorithm has four steps:

1. For n generating protoconcepts $\mathfrak{p}_1, \dots, \mathfrak{p}_n$ we set $\widetilde{M}_1 := \{\mathfrak{p}_{1\sqcup}, \dots, \mathfrak{p}_{n\sqcup}\}$, $\widetilde{G}_1 := \{\mathfrak{p}_{1\sqcap}, \dots, \mathfrak{p}_{n\sqcap}\}$ and $M_0 := G_0 := \emptyset$.
2. (a) For $i \geq 1$ we determine the set

$$M_i := \{(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \text{ is a full } \sqcup\text{-sequent over } \widetilde{M}_i \text{ and } j(\mathfrak{A}, \mathfrak{B}) \neq \top\}.$$

by asking the user which of the implications $\sqcap \mathfrak{A} \rightarrow \sqcup \mathfrak{B}$ are true.

- (b) Dually we determine for $i \geq 1$ the set

$$G_i := \{(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \text{ is a full } \sqcap\text{-sequent over } \widetilde{G}_i \text{ and } m(\mathfrak{A}, \mathfrak{B}) \neq \perp\}.$$

3. We set $\mathbb{K}_i := (m(G_i), j(M_i), \sqsubseteq)$, where we determine the relation \sqsubseteq with the aid of the expert.
4. We stop if $|G_{i-1}| = |G_i|$ and $|M_{i-1}| = |M_i|$, otherwise we set $\widetilde{G}_{i+1} := \widetilde{G}_i \cup \{(j(\mathfrak{A}, \mathfrak{B}))_{\sqcap} \mid (\mathfrak{A}, \mathfrak{B}) \in M_i\}$ and $\widetilde{M}_{i+1} := \widetilde{M}_i \cup \{(m(\mathfrak{A}, \mathfrak{B}))_{\sqcup} \mid (\mathfrak{A}, \mathfrak{B}) \in G_i\}$ and continue with 2).

Proposition 1 makes clear that in every step of the algorithm the sequents in M_i describe the blocks of the partition $\pi_{\widetilde{M}_i}$ of M , and, dually, the sequents of G_i describe the blocks of the partition $\pi_{\widetilde{G}_i}$ of G . Therefore, we can take $m(G_i)$ as objects and $j(M_i)$ as the attributes of the contexts \mathbb{K}_i . The next lemmas and propositions prepare Theorem 3 and Theorem 4 where the correctness of the algorithm is proven. Thereafter, Proposition 4 explains the relation between the contexts \mathbb{K}_i and, finally, a brief example is given.

Lemma 1. 1) Let $\mathfrak{p} \in \widetilde{M}_i$. Then $\mathfrak{p}_{\sqcup} = \sqcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in M_i \text{ and } \mathfrak{p} \in \mathfrak{B}\}$ and $\neg \mathfrak{p}_{\sqcup} = \sqcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in M_i \text{ and } \mathfrak{p} \in \mathfrak{A}\}$
 2) Dually, let $\mathfrak{p} \in \widetilde{G}_i$. Then $\mathfrak{p}_{\sqcap} = \sqcup \{m(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in G_i \text{ and } \mathfrak{p} \in \mathfrak{A}\}$ and $\neg \mathfrak{p}_{\sqcap} = \sqcup \{m(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in G_i \text{ and } \mathfrak{p} \in \mathfrak{B}\}$

Proof: This is a consequence of Proposition 2. For 1) we have that $\mathfrak{p}_{\sqcup} = j(\emptyset, \{\mathfrak{p}\})$ and thus $\mathfrak{p}_{\sqcup} = \sqcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in M_i \text{ and } (\emptyset, \{\mathfrak{p}\}) \leq (\mathfrak{A}, \mathfrak{B})\} = \sqcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in M_i \text{ and } \mathfrak{p} \in \mathfrak{B}\}$. Analogously it follows from $\neg \mathfrak{p}_{\sqcup} = j(\{\mathfrak{p}\}, \emptyset)$ that $\neg \mathfrak{p}_{\sqcup} = \sqcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in M_i \text{ and } \mathfrak{p} \in \mathfrak{A}\}$. Dually we obtain 2). \square

The next lemma ensures that the protoconcepts of $j(M_i)$ and $m(G_i)$ can be obtained from the elements of $j(M_k)$ and $m(G_k)$ in the following iterations ($k \geq i$).

Lemma 2. 1) If $(\mathfrak{A}, \mathfrak{B}) \in M_i$ then we can find a set $S \subseteq M_{i+1}$ such that $j(\mathfrak{A}, \mathfrak{B}) = \bigcap \{j(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \in S\}$.

2) Dually, if $(\mathfrak{A}, \mathfrak{B}) \in G_i$ then we can find a set $S \subseteq G_{i+1}$ such that $m(\mathfrak{A}, \mathfrak{B}) = \bigcup \{m(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \in S\}$.

Proof: Since $\widetilde{M}_i \subseteq \widetilde{M}_{i+1}$, a \sqcup -sequent $(\mathfrak{A}, \mathfrak{B})$ over \widetilde{M}_i is also a \sqcup -sequent over \widetilde{M}_{i+1} . For $S = \{(\mathfrak{C}, \mathfrak{D}) \in M_{i+1} \mid (\mathfrak{A}, \mathfrak{B}) \leq \mathfrak{C}, \mathfrak{D}\}$ we conclude from Proposition 2 that $j(\mathfrak{A}, \mathfrak{B}) = \bigcap \{j(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \in S\}$. The second part is obtained dually. \square

Proposition 3. 1) If for a \sqcap -semiconcept \mathfrak{p} there is a set $S \subseteq G_i$ such that $\mathfrak{p} = \bigcup \{m(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S\}$ then we can find a set $T \subseteq M_{i+1}$ such that $\mathfrak{p}_{\sqcup} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in T\}$.

2) Dually, if for a \sqcup -semiconcept \mathfrak{p} there is a set $S \subseteq M_i$ such that $\mathfrak{p} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S\}$ then we can find a set $T \subseteq G_{i+1}$ such that $\mathfrak{p}_{\sqcap} = \bigcup \{m(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in T\}$.

Proof: 1) Since for semiconcepts $\mathfrak{x}, \mathfrak{y}$ $(\mathfrak{x} \sqcup \mathfrak{y})_{\sqcup} = \mathfrak{x} \sqcup \mathfrak{y}$ (cf. [Vo03]), it follows from $\mathfrak{p} = \bigcup \{m(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S\}$ and from $(\mathfrak{p} \sqcup \mathfrak{y})_{\sqcup} = \mathfrak{p} \sqcup \mathfrak{y}$ that $\mathfrak{p}_{\sqcup} = \bigcap \{(m(\mathfrak{A}, \mathfrak{B}))_{\sqcup} \mid (\mathfrak{A}, \mathfrak{B}) \in S\}$. Since $(m(\mathfrak{A}, \mathfrak{B}))_{\sqcup} \in \widetilde{M}_{i+1}$ for $(\mathfrak{A}, \mathfrak{B}) \in S$, Lemma 1 yields $(m(\mathfrak{A}, \mathfrak{B}))_{\sqcup} = \bigcap \{j(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \in M_{i+1} \text{ and } (m(\mathfrak{A}, \mathfrak{B}))_{\sqcup} \in \mathfrak{D}\}$. We conclude from Corollary 1 that

$$\begin{aligned} \mathfrak{p}_{\sqcup} &= \bigcap_{(\mathfrak{A}, \mathfrak{B}) \in S} (\bigcap \{j(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \in M_{i+1} \text{ and } (m(\mathfrak{A}, \mathfrak{B}))_{\sqcup} \in \mathfrak{D}\}) \\ &= \bigcap \{j(\mathfrak{C}, \mathfrak{D}) \mid (\mathfrak{C}, \mathfrak{D}) \in M_{i+1} \text{ and } (m(\mathfrak{A}, \mathfrak{B}))_{\sqcup} \in \mathfrak{D} \text{ for every } (\mathfrak{A}, \mathfrak{B}) \in S\}. \end{aligned}$$

2) follows dually. \square

In order to describe how much of \mathfrak{P}_g^p the algorithm has found after i iterations we need to measure how difficult a protoconcept is to generate from the set of basic protoconcepts \mathcal{B} . Note that if T is the set of all terms over a set of variables $\{x_1, \dots, x_n\}$ with the operations $\sqcup, \sqcap, \sqcup, \sqcap, \neg, \neg, \top, \perp$, and if \mathfrak{P} is a protoconcept algebra, and $\mathcal{B} = \{\mathfrak{p}_1, \dots, \mathfrak{p}_n\} \subseteq \mathfrak{P}$ then there is a unique homomorphism $\phi : T \rightarrow \mathfrak{P}$ with $\phi(x_i) = \mathfrak{p}_i$.

Definition 5. For a term t over the set $X = \{x_1, \dots, x_n\}$ we define the depth $d(t)$ of t by:

- $d(x_i) = 1$, for $i \in \{1, \dots, n\}$
- $d(\perp) = d(\top) = 1$
- $d(t_1 \sqcap t_2) = d(t_1 \sqcup t_2) = d(t_1 \sqcap t_2) = d(t_1 \sqcup t_2) = \max \{d(t_1), d(t_2)\} + 1$ for terms t_1, t_2
- $d(\neg t) = d(\neg t) = d(t) + 1$ for a term t .

We say that a protoconcept \mathfrak{p} of a protoconcept algebra \mathfrak{P} is generated in m steps from the protoconcepts $\mathfrak{p}_1, \dots, \mathfrak{p}_n$ in \mathfrak{P} iff there is a term t over X with $d(t) \leq m$ and $\phi(t) = \mathfrak{p}$ in \mathfrak{P} .

The following theorem is the main result of this paper. It shows which part of \mathfrak{P}_g^p is computed by the algorithm after i iterations.

Theorem 3. *Let \mathbf{p} be a semiconcept generated by $\mathcal{B} := \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ in \mathfrak{P}_u in i steps. If \mathbf{p} is a \sqcup -semiconcept then there exists a subset $A \subseteq M_m$ such that $\mathbf{p} = \bigcap j(A)$. Dually, if \mathbf{p} is a \sqcap -semiconcept then there exists a subset $A \subseteq G_m$ such that $\mathbf{p} = \bigcup m(A)$.*

Proof: We use induction over i . For $i = 1$ and a \sqcup -semiconcept \mathbf{p} we have that $\mathbf{p} = \top$ or $\mathbf{p} \in \mathcal{B}$. If $\mathbf{p} = \top$ then $\mathbf{p} = \bigcap \emptyset$. If $\mathbf{p} = \mathbf{p}_j$ for some $j \in \{1, \dots, n\}$ then the fact that $\mathbf{p} = \mathbf{p}_j = \mathbf{p}_{j\sqcup} \in \widetilde{M}_1$ and Lemma 1 imply that there exists a set $A \subseteq M_1$ such that $\mathbf{p} = \bigcap j(A)$. Dually, we see that for a \sqcap -semiconcept \mathbf{p} that can be generated in one step we find a set $A \subseteq G_1$ such that $\mathbf{p} = \bigcup m(A)$.

For the conclusion from i to $i+1$ we assume that \mathbf{p} is a \sqcup -semiconcept which can be generated in \mathfrak{P}_u from \mathcal{B} in $i+1$ steps. If \mathbf{p} can also be generated in i steps, we obtain from Lemma 2 that we can find such a set $S \subseteq M_{i+1}$. If \mathbf{p} cannot be generated faster, then there exist semiconcepts \mathbf{p}_x and \mathbf{p}_y such that a) $\mathbf{p} = \neg \mathbf{p}_x$, b) $\mathbf{p} = \mathbf{p}_x \sqcap \mathbf{p}_y$ or c) $\mathbf{p} = \mathbf{p}_x \sqcup \mathbf{p}_y$, where \mathbf{p}_x and \mathbf{p}_y can be generated in i steps.

a) As \mathbf{p}_x can be generated in i steps, there is either a set $S_M \subseteq M_i$ with $\mathbf{p}_x = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_M\}$ or a set $S_G \subseteq G_i$ with $\mathbf{p}_x = \bigcup \{m(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_G\}$. In the first case, Proposition 1 yields that the intent of \mathbf{p}_x is a union of blocks of the partition $\pi_{\widetilde{M}_i}$. Therefore the complement of the intent is the union of all other blocks of $\pi_{\widetilde{M}_i}$ and thus $\mathbf{p} = \neg \mathbf{p}_x = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in M_i \setminus S_M\}$. Lemma 2 then yields that we can also find a set $A \subseteq M_{i+1}$ such that $\mathbf{p} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in A\}$.

In the second case, $\mathbf{p}_x = \bigcup \{m(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_G\}$, Proposition 3 ensures that we can find a set $T \subseteq M_{i+1}$ such that $\mathbf{p}_{x\sqcup} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in T\}$. Again we obtain $\neg \mathbf{p}_x = \neg \mathbf{p}_{x\sqcup} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in M_{i+1} \setminus T\}$.

b) Since the semiconcepts \mathbf{p}_x and \mathbf{p}_y can be generated in i steps we find again with Lemma 2 or Proposition 3 sets S_x and S_y in M_{i+1} such that $\mathbf{p}_{x\sqcup} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_x\}$ and $\mathbf{p}_{y\sqcup} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_y\}$. Then $\mathbf{p} = \mathbf{p}_{x\sqcup} \sqcap \mathbf{p}_{y\sqcup} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_x\} \sqcap \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_y\} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_x \cup S_y\}$ and $A := S_x \cup S_y$ is the set we need.

c) Similar to b) we find sets S_x and S_y as above and Corollary 1 yields $\mathbf{p} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_x\} \sqcup \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_y\} = \bigcap \{j(\mathfrak{A}, \mathfrak{B}) \mid (\mathfrak{A}, \mathfrak{B}) \in S_x \cap S_y\}$. Dually, we obtain a set $A \subseteq G_{i+1}$ for \sqcap -semiconcepts. \square

Theorem 4. *If the algorithm stops after i iterations, then $\mathfrak{H}(\mathbb{K}_i) \cup \mathcal{B}$ is the subalgebra \mathfrak{P}_g of \mathfrak{P}_u generated by \mathcal{B} .*

Proof: We stop if $|G_{i-1}| = |G_i|$ and $|M_{i-1}| = |M_i|$. From $\widetilde{G}_{i-1} \subseteq \widetilde{G}_i$ it follows that $\pi_{\widetilde{G}_i} \leq \pi_{\widetilde{G}_{i-1}}$. From $|G_{i-1}| = |G_i|$ we conclude that the number of blocks of the partitions is equal, therefore the partitions must be equal, hence $m(G_{i-1}) = m(G_i)$. Likewise $j(M_{i-1}) = j(M_i)$ and we cannot generate any new protoconcepts, i.e. $m(G_i) = m(G_n)$, $j(M_i) = j(M_n)$ and $\mathbb{K}_i = \mathbb{K}_n$ for $n \geq i$.

The previous theorem then yields that $\mathfrak{H}(\mathbb{K}_i)$ is the set of all semiconcepts that can be generated from \mathcal{B} and thus $\mathfrak{P}_g = \mathfrak{H}(\mathbb{K}_i) \cup \mathcal{B}$. \square

The following proposition describes how the contexts \mathbb{K}_i develop during the algorithm.

Proposition 4. *Let $i, j \in \mathbb{N}$, $i \leq j$ and let I_i, I_j be the incidence relations of the contexts $\mathbb{K}_i, \mathbb{K}_j$. We define surjective maps $\gamma_{ji} : m(G_j) \rightarrow m(G_i)$ and $\mu_{ji} : j(M_j) \rightarrow j(M_i)$ by*

$$\begin{aligned}\gamma_{ji}(m(\mathfrak{A}, \mathfrak{B})) &= m(\mathfrak{A} \cap \widetilde{G}_i, \mathfrak{B} \cap \widetilde{G}_i) \\ \mu_{ji}(j(\mathfrak{A}, \mathfrak{B})) &= j(\mathfrak{A} \cap \widetilde{M}_i, \mathfrak{B} \cap \widetilde{M}_i).\end{aligned}$$

Then for $g \in m(G_i)$ and $m \in j(M_i)$

$$gI_im \Leftrightarrow \gamma_{ji}^{-1}(g)I_j\mu_{ji}^{-1}(m).$$

Proof: Since for $j \geq i$ holds $\widetilde{G}_i \subseteq \widetilde{G}_j$, $(\mathfrak{A} \cap \widetilde{G}_i, \mathfrak{B} \cap \widetilde{G}_i)$ is a full sequent over \widetilde{G}_i . Moreover, from $m(\mathfrak{A}, \mathfrak{B}) \neq \perp$ it follows that $m(\mathfrak{A} \cap \widetilde{G}_i, \mathfrak{B} \cap \widetilde{G}_i) \neq \perp$, hence $(\mathfrak{A} \cap \widetilde{G}_i, \mathfrak{B} \cap \widetilde{G}_i) \in G_i$ and γ_{ji} is well defined. Since π_{M_j} is a refinement of π_{M_i} , γ_{ji} is surjective. Dually μ_{ji} . For $(\mathfrak{A}_1, \mathfrak{B}_1) \in G_j$ and $(\mathfrak{A}_2, \mathfrak{B}_2) \in M_j$ it follows from $m(\mathfrak{A}_1 \cap \widetilde{G}_i, \mathfrak{B}_1 \cap \widetilde{G}_i) I_i j(\mathfrak{A}_2 \cap \widetilde{M}_i, \mathfrak{B}_2 \cap \widetilde{M}_i)$ that $m(\mathfrak{A}_1, \mathfrak{B}_1) \sqsubseteq m(\mathfrak{A}_1 \cap \widetilde{G}_i, \mathfrak{B}_1 \cap \widetilde{G}_i) \sqsubseteq j(\mathfrak{A}_2 \cap \widetilde{M}_i, \mathfrak{B}_2 \cap \widetilde{M}_i) \sqsubseteq j(\mathfrak{A}_2, \mathfrak{B}_2)$, hence $gI_im \Rightarrow \gamma_{ji}^{-1}(g)I_j\mu_{ji}^{-1}(m)$. Conversely, if $(\mathfrak{A}_1, \mathfrak{B}_1) \in G_i$, $(\mathfrak{A}_2, \mathfrak{B}_2) \in M_i$ and $\gamma_{ji}^{-1}(m(\mathfrak{A}_1, \mathfrak{B}_1))I_j\mu_{ji}^{-1}(j(\mathfrak{A}_2, \mathfrak{B}_2))$, then $m(\mathfrak{A}_1, \mathfrak{B}_1) = \bigsqcup \gamma_{ji}^{-1}(m(\mathfrak{A}_1, \mathfrak{B}_1)) \sqsubseteq \bigsqcup \mu_{ji}^{-1}(j(\mathfrak{A}_2, \mathfrak{B}_2)) = j(\mathfrak{A}_2, \mathfrak{B}_2)$, hence $m(\mathfrak{A}_1, \mathfrak{B}_1) I_i j(\mathfrak{A}_2, \mathfrak{B}_2)$. \square

In the following a small example is given. As the algorithm is redundant, we shorten some steps and give ideas for improvement. Moreover, due to space limitations, only the first two iterations are presented. These first steps are sufficient to illustrate the algorithm, while in later steps the implications become very complex.

Example. We want to explore a context of inland waters starting from an intuitive understanding of “river” as an inland water and “artificial” as an attribute describing inland waters. As basic protoconcepts we take thus the \sqcap -semiconcept $(\{river\}, \{river\}')$ generated by “river” which we abbreviate by γr and the \sqcup -semiconcept $(\{artificial\}', \{artificial\})$ generated by “artificial” which we abbreviate by μa . For the first iteration of the algorithm we set in step 1: $\widetilde{M}_1 := \{\gamma r_{\sqcup}, \mu a\}$ and $\widetilde{G}_1 := \{\gamma r, \mu a_{\sqcap}\}$. In step 2, the user is asked which of the following implications between \sqcup -semiconcepts are true:

Implication	true	Interpretation
$\top \rightarrow \mu a \sqcup \gamma r_{\sqcup}$	yes	The intents of γr_{\sqcup} and μa are disjoint
$\gamma r_{\sqcup} \sqcap \mu a \rightarrow \perp$	no	Every attribute is an attribute of “river” or “artificial”
$\gamma r_{\sqcup} \rightarrow \mu a$	no	“artificial” is an attribute of “river”
$\mu a \rightarrow \gamma r_{\sqcup}$	no	the attributes of “river” form a subset of “artificial”

Likewise, the user has to decide which of the following implications between \sqcap -semiconcepts are true:



Implication	true	Interpretation
$\gamma r \sqcap \mu a_{\sqcap} \rightarrow \perp$	yes	The extents of γr and μa_{\sqcap} are disjoint
$\gamma r \rightarrow \mu a_{\sqcap}$	no	“river” is an artificial water
$\mu a_{\sqcap} \rightarrow \gamma r$	no	The artificial waters form a subset of “river”
$\top \rightarrow \mu a_{\sqcap} \sqcup \gamma r$	no	Every inland water is artificial or a river

We obtain

$$M_1 = \{(\{\mu a, \gamma r_{\sqcup}\}, \emptyset), (\{\gamma r_{\sqcup}\}, \{\mu a\}), (\{\mu a\}, \{\gamma r_{\sqcup}\})\},$$

$$G_1 = \{(\{\gamma r\}, \{\mu a_{\sqcap}\}), (\{\mu a_{\sqcap}\}, \{\gamma r\}), (\emptyset, \{\gamma r, \mu a_{\sqcap}\})\}.$$

From the true implication $\top \rightarrow \mu a \sqcup \gamma r$ follows $j(\{\gamma r\}, \{\mu a\}) = j(\emptyset, \{\mu a\})$ and $j(\{\mu a\}, \{\gamma r\}) = j(\emptyset, \{\gamma r\})$. Likewise $\gamma r \sqcap \mu a_{\sqcap} \rightarrow \perp$ yields $m(\{\gamma r\}, \{\mu a_{\sqcap}\}) = m(\{\gamma r\}, \emptyset)$ and $m(\{\mu a_{\sqcap}\}, \{\gamma r\}) = m(\{\mu a_{\sqcap}\}, \emptyset)$. In order to improve readability we use the smaller sequents. Thus we obtain the context \mathbb{K}_1 as:

	$\neg \gamma r_{\sqcup} \sqcup \mu a$	$\neg \mu a_{\sqcap}$	γr_{\sqcup}
γr			
μa_{\sqcap}			
$\neg \mu a_{\sqcap}$ $\neg \gamma r$			

Since the \sqcap -semiconcepts are defined by their extent, it is sufficient to explain the extents of the objects of \mathbb{K}_1 . The extent of γr is obviously “river”, the extent of μa_{\sqcap} is the set of all artificial inland waters, and the extent of $\neg \mu a_{\sqcap} \sqcap \neg \gamma r$ is the set of all natural inland waters except “river”. The intent of $\neg \gamma r_{\sqcup} \sqcup \neg \mu a$ is the set of all attributes “river” does not have from which furthermore “artificial” is excluded, the intent of μa is “artificial”, and the intent of γr_{\sqcup} is the set of all attributes “river” has. Finally, we set $\tilde{M}_2 := \{\gamma r_{\sqcup}, \mu a, \mu a_{\sqcap}, (\neg \mu a_{\sqcap} \sqcap \neg \gamma r)_{\sqcup}\}$ and $\tilde{G}_2 := \{\gamma r, \mu a_{\sqcap}, \gamma r_{\sqcup}, (\neg \gamma r \sqcup \neg \mu a)_{\sqcap}\}$.

For the second iteration, we define a *prime implicand* as a minimal sequent describing a true implication. In order to save space, instead of all full sequents a list of the implications described by prime implicands is given. For \sqcup -sequents we have:

Implication	Interpretation
$\mu a \rightarrow \mu a_{\sqcap}$	“artificial” implies no other attributes
$\mu a_{\sqcap} \rightarrow \mu a$	always true
$\top \rightarrow \mu a_{\sqcap} \sqcup \gamma r_{\sqcup}$	follows from the other true implications
$\top \rightarrow (\neg \mu a \sqcap \neg \gamma r)_{\sqcup} \sqcup \mu a$	There are natural waters that are not rivers
$\top \rightarrow (\neg \mu a \sqcap \neg \gamma r)_{\sqcup} \sqcup \mu a_{\sqcap}$	follows from the rest
$\gamma r_{\sqcup} \rightarrow (\neg \mu a \sqcap \neg \gamma r)_{\sqcup}$	“river” has all attributes common to all natural waters










For \sqcap -sequents we obtain:

Implication	Interpretation
$\gamma r \rightarrow \gamma r \sqcup \sqcap$	always true
$\gamma r \sqcup \sqcap \rightarrow \gamma r$	The attributes of “river” distinguish it from all other waters
$\gamma r \sqcup \sqcap \sqcap \mu a \rightarrow \perp$	Follows from the other implications
$(\neg \gamma r \sqcup \neg \mu a) \sqcap \sqcap \gamma r \rightarrow \perp$	There are more attributes than “artificial” that “river” does not have
$(\neg \gamma r \sqcup \neg \mu a) \sqcap \sqcap \gamma r \sqcup \sqcap \rightarrow \perp$	Follows from the other implications

Reducing the full sequents as in the first iteration then yields

$$\begin{aligned}
 M_2 &:= \{(\{\gamma r \sqcup, \mu a\}, \emptyset), (\emptyset, \{\mu a\}), (\emptyset, \{\gamma r \sqcup, (\neg \mu a \sqcap \neg \gamma r) \sqcup\}), \\
 &\quad (\{\neg \mu a \sqcap \neg \gamma r\} \sqcup, \{\gamma r\})\}, \\
 G_2 &:= \{(\{\gamma r\}, \emptyset), (\{\mu a, (\neg \gamma r \sqcup \neg \mu a) \sqcap\}, \emptyset), (\{\mu a\}, \{(\neg \gamma r \sqcup \neg \mu a) \sqcap\}), \\
 &\quad (\{(\neg \gamma r \sqcup \neg \mu a) \sqcap\}, \{\mu a \sqcap\}), (\emptyset, \{(\neg \gamma r \sqcup \neg \mu a) \sqcap, \mu a \sqcap, \gamma r\})\}.
 \end{aligned}$$

The extent of the semiconcept described by $(\neg \gamma r \sqcup) \sqcap$ is the set of all inland waters that have all attributes “river” does not have. We understand this as the “opposite” of “river” and decide that in our understanding of the context of inland waters this opposite is the object “basin”. In the same way we decide that the intent of the semiconcept $(\neg \mu a \sqcap) \sqcup$ is the attribute “natural”. Moreover, we decide that as inland waters similar to “basin” but natural (i.e. it has all the attributes “river” does not have except for “artificial”) we have only “pond”. Therefore, in \mathbb{K}_2 we replace $m(\{\gamma r\}, \emptyset)$ with “river”, $m(\{\mu a\}, \{(\neg \gamma r \sqcup \neg \mu a) \sqcap\})$ with “basin”, $m(\{(\neg \gamma r \sqcup \neg \mu a) \sqcap\}, \{\mu a \sqcap\})$ with “pond”, $j(\emptyset, \{\mu a\})$ with “artificial” and $j(\emptyset, \{\gamma r \sqcup, (\neg \mu a \sqcap \neg \gamma r) \sqcup\})$ with “natural”. We obtain the context \mathbb{K}_2 as:

\mathbb{K}_2	$\neg \mu a \sqcap \neg \gamma r \sqcup$	“artificial”	“natural”	$\neg(\neg \mu a \sqcap \neg \gamma r) \sqcup \gamma r \sqcup$
“river”				
“basin”				
$\mu a \sqcap \neg(\neg \gamma r \sqcup \neg \mu a) \sqcap$				
“pond”				
$\neg(\neg \gamma r \sqcup \neg \mu a) \sqcap$				
$\sqcap \neg \mu a \sqcap \sqcap \neg \gamma r$				

The extent of $\mu a \sqcap \neg(\neg \gamma r \sqcup \neg \mu a) \sqcap$ is the set of all artificial waters g for which an attribute $m \in M \setminus \text{river}'$ exists such that $m \notin g'$. The extent of $\neg(\neg \gamma r \sqcup$

$\neg\mu a)_{\sqcap} \sqcap \neg\mu a_{\sqcap} \sqcap \neg\gamma r$ is the set of all natural waters g for which an attribute $m \in M \setminus \text{river}'$ exists such that $m \notin g'$. The intent of $\neg\mu a_{\sqcap} \sqcap \neg\gamma r_{\sqcup}$ is the set of all attributes that “river” does not from which furthermore “artificial” is excluded, and the intent of $\neg(\neg\mu a_{\sqcap} \sqcap \neg\gamma r) \sqcup \gamma r_{\sqcup}$ is the set of all attributes that “river” has and that are not common to all natural waters.

We stop here due to space restrictions. In the third iteration the attribute set would be increased by one, then the algorithm terminates.

5 Further Research

As mentioned before, the presented algorithm still needs to be optimized. Some possibilities are shown in the example. Firstly, it is helpful to allow the user to assign names to the elements of G_i and M_i in every step and to use the known implications to reduce the full sequents of G_i and M_i . Secondly, the set of all valid or invalid full sequents is highly redundant. This can be improved by determining the prime implicands with the aid of the user. Then the sets M_i and G_i could be computed from the prime implicands. Still, a strategy to determine these sequents rapidly and avoiding redundancies is needed. Moreover, the incidence relation of the \mathbb{K}_i could be used to find valid implications and Proposition 4 describes how it is related to the incidence relation of \mathbb{K}_{i+1} . Finally, a further investigation of double Boolean algebras in order to determine the free double Boolean algebras is crucial.

References

- [GW99] B. Ganter, R. Wille: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg 1999.
- [Gr68] G. Grätzer: Universal Algebra. van Nostrand, Princeton, 1968.
- [HLSW00] C. Herrman, P. Luksch, M. Skorsky, R. Wille: Algebras of Semiconcepts and Double Boolean Algebras. *Contributions to General Algebra 13*. Verlag Johannes Heyn, Klagensfurt 2000, 175–188.
- [KV03] J. Klinger, B. Vormbrock: Contextual Boolean Logic: How did it develop? In: B. Ganter, A. de Moor (eds.): *Using Conceptual Structures. Contributions to ICCS 2003*. Shaker, Aachen 2003, 143–156.
- [St97] G. Stumme: Concept Exploration. Knowledge Acquisition in Conceptual Knowledge Systems. Shaker, Aachen 1997.
- [Vo03] B. Vormbrock: Congruence Relations on Double Boolean Algebras. FB4-Preprint 2287, TU Darmstadt 2003.
- [VW03] B. Vormbrock, R. Wille: Semiconcept and Protoconcept Algebras: The Basic Theorems. FB4-Preprint, TU Darmstadt 2003.
- [Wi00a] R. Wille: Boolean Concept Logic. In: B. Ganter, G. W. Mineau (eds.): *Conceptual structures: logical, linguistic, and computational issues*. LNAI 1867. Springer, Heidelberg 2000, 317–331.
- [Wi00b] R. Wille: *Contextual Logic Summary*. In: G. Stumme (Ed.): Working with Conceptual Structures. Contributions to ICCS 2000. Shaker, Aachen 2000, 265–276.

Geometry of Data Tables

Tim B. Kaiser^{1,2} and Stefan E. Schmidt²

¹ Department of Mathematics, Darmstadt University of Technology
64289 Darmstadt, Germany

`tkaiser@mathematik.tu-darmstadt.de`

² Physical Science Laboratory, New Mexico State University
88001 Las Cruces, USA
`schmidt@nmsu.edu`

Abstract. We introduce the notions of systems of equivalence relations, affine ordered sets and projective ordered sets to draw connections between many-valued contexts, geometry, and order-theory. This gives rise to applications in data analysis and data visualization.

1 Introduction

A data table, formalized as a (*complete*) *many-valued context*, can be captured by a *system of equivalence relations*. Section 2 provides the formalizations and definitions necessary to express this connection. The *geometry* of a data table consists of the ordered set of all equivalence classes of all equivalence relations of the corresponding system of equivalence relations. In our approach we label each equivalence class with the relation it stems from (compare [Wi70] where congruence classes are ordered only via set-inclusion, which can result in identifying classes from different relations). This ordered set has special properties which are reflected in the notion of an *affine ordered set*. The latter carries a parallelism which allows the reconstruction of the data table. The notion of a *projective ordered set* goes further. Here the ordered set is equipped with a specified subset of so-called *infinity objects*. For weaker versions of both concepts, *ordered sets with parallelism* and *weakly projective ordered sets*, there exists a categorical equivalence, which is explained in Section 3. The categorical equivalence between affine ordered sets and systems of equivalence relations with identity relation is established in Section 4. Section 5 summarizes existing categorical equivalences and Section 6 shows a real data example to indicate possible applications.

2 Simple Many-Valued Contexts and Systems of Equivalence Relations

Data tables are formalized as many-valued contexts as it is common for example in Formal Concept Analysis [GW99]. Many-valued contexts are also known as *Chu Spaces* [Pr95] or *Knowledge Representation Systems* [Pa91].

Definition 1 (many-valued context). A (*complete*) many-valued context is a structure $\mathbb{K} := (G, M, W, I)$, where G is a set of objects, M is a set of attributes,

W is a set of values and $I \subseteq G \times M \times W$ is a ternary relation, where for every $(g, m) \in G \times M$ there exists a unique $w \in W$ with $(g, m, w) \in I$ (i.e. m is a function from G to W via $m(g) := w$).

We call an attribute $m \in M$ an *id attribute* if for any two objects $g_1, g_2 \in G$ the values of g_1 and g_2 regarding to m are different (i.e. $m(g_1) \neq m(g_2)$). The following definition provides a notion of dependency between attributes of a many-valued context.

Definition 2 (functional dependence). *If $m_1, m_2 \in M$ are attributes of a many-valued context (G, M, W, I) , we call m_2 functionally dependent upon m_1 (in symbols: $m_1 \subseteq m_2$) if*

$$\forall g, h \in G : m_1(g) = m_1(h) \Rightarrow m_2(g) = m_2(h).$$

If $m_1 \subseteq m_2$ and $m_2 \subseteq m_1$, the attributes m_1 and m_2 are called functionally equivalent.

For a map $f : A \rightarrow B$ the *kernel* of f is defined as the equivalence relation $\ker(f) := \{(a, b) \in A^2 \mid f(a) = f(b)\}$. It is easily seen that $m_1 \subseteq m_2$ if and only if $\ker(m_1) \subseteq \ker(m_2)$. Accordingly, m_1 and m_2 are functionally equivalent if and only if $\ker(m_1) = \ker(m_2)$.

Note that our definition of functional dependence is a restricted one, since it allows only one-element premises.

We can reduce a given many-valued context by identifying functionally equivalent attributes. This is reasonable because there exists always a bijection $f : m_1(G) \rightarrow m_2(G)$ between the values of functionally equivalent attributes with $f(m_1(g)) = m_2(g)$ for all $g \in G$.

Many-valued contexts where any two functionally equivalent attributes are equal will be called *simple*. Every many-valued context can be simplified without loss of information via identifying functionally equivalent attributes.

Definition 3 (homomorphism for many-valued contexts). *For many-valued contexts (G, M, W, I) and (G_0, M_0, W_0, I_0) we call a pair of mappings (α, β) with $\alpha : G \rightarrow G_0$ and $\beta : M \rightarrow M_0$ a homomorphism between the many-valued contexts if*

1. $\forall g, h \in G : m(g) = m(h) \implies \beta(m)(\alpha(g)) = \beta(m)(\alpha(h))$ and
2. β preserves functional dependence, that is

$$\forall m_1, m_2 \in M : m_1 \subseteq m_2 \implies \beta(m_1) \subseteq \beta(m_2).$$

We can rephrase the conditions in Definition 3 as

1. $\forall g, h \in G : (g, h) \in \ker(m) \implies (\alpha(g), \alpha(h)) \in \ker(\beta(m))$ and
2. $\forall m_1, m_2 \in M : \ker(m_1) \subseteq \ker(m_2) \implies \ker(\beta(m_1)) \subseteq \ker(\beta(m_2))$.

To link a simple many-valued context (G, M, W, I) to geometry we consider the family of equivalence relations $(\ker(m))_{m \in M}$ on G . Then, every attribute

$m \in M$ induces a partition on the object set via the equivalence classes of $\ker(m)$. So we can regard a simple many-valued context as a set of partitions induced by its attributes. Every block of a partition corresponds to the set of objects with a certain value with respect to a certain attribute. We denote the *identity relation* on the set X by $\Delta_X := \{(x, x) \mid x \in X\}$.

Definition 4 (system of equivalence relations). We call $\mathbb{E} := (D, E)$ a system of equivalence relations (SER), if D is a set and E is a set of equivalence relations on D . If $d \in D$ and $\theta \in E$, we denote the equivalence class of d by $[d]\theta := \{d' \in D \mid d'\theta d\}$. If $\Delta_D \in E$ we will also call (D, E) an “SER with identity relation”.

Definition 5 (homomorphism for SER). For SERs (D, E) and (D_0, E_0) we call a pair of mappings (α, β) with $\alpha : D \rightarrow D_0$ and $\beta : E \rightarrow E_0$ a homomorphism between the SERs if

1. $\forall x, y \in D \forall \theta \in E: x\theta y \Rightarrow \alpha(x)\beta(\theta)\alpha(y)$ and
2. $\forall \theta_1, \theta_2 \in E: \theta_1 \subseteq \theta_2 \Rightarrow \beta(\theta_1) \subseteq \beta(\theta_2)$.

To every given SER $\mathbb{E} := (D, E)$ we can assign a simple many-valued context $\mathbf{K}(\mathbb{E}) := (D, E, W, I)$, where $W := \{[d]\theta \mid d \in D, \theta \in E\}$ and $I := \{(d, \theta, w) \in D \times E \times W \mid w = [d]\theta\}$.

On the other hand we can assign, as described above, a SER to every many-valued context. We define $\mathbf{E}(\mathbb{K}) := (G, \{\ker(m) \mid m \in M\})$.

We observe that for every simple many-valued context \mathbb{K} we have $\mathbf{K}(\mathbf{E}(\mathbb{K})) \simeq \mathbb{K}$ and for every SER \mathbb{E} we have $\mathbf{E}(\mathbf{K}(\mathbb{E})) = \mathbb{E}$.

The equivalence classes of the partitions form naturally an ordered set with special properties. This observation enables us to approach many-valued contexts using geometric and order-theoretic concepts and methods. The next chapter deals with the relationship between two order-theoretic concepts linked closely to geometry: ordered sets with parallelism and weakly projective ordered sets. These concepts are equivalent to SERs with identity relation and many-valued contexts with id attribute if specialized appropriately. This is elaborated in Sections 4 and 5.

3 Ordered Sets with Parallelism and Weakly Projective Ordered Sets

In this section we introduce the categories of *ordered sets with parallelism* and *weakly projective ordered sets* and establish a categorical equivalence between them.

Definition 6 (ordered set with parallelism). We call a triple $\mathbb{A} := (Q, \leq, \parallel)$ an ordered set with parallelism, if (Q, \leq) is a partially ordered set, \parallel is a equivalence relation on Q , and the axioms (A1) - (A3) hold. Let $A := \text{Min}(Q, \leq)$ denote the set of all minimal elements in (Q, \leq) and $A(x) := \{a \in A \mid a \leq x\}$.

(A1) $\forall x \in Q : A(x) \neq \emptyset$

(A2) $\forall x \in Q \forall a \in A \exists! t \in Q : a \leq t \parallel x$

(A3) $\forall x, y, x', y' \in Q : x' \parallel x \leq y \parallel y' \ \& \ A(x') \cap A(y') \neq \emptyset \Rightarrow x' \leq y'$

The elements of A are called points and, in general, elements of Q are called subspaces. We say that a subspace x is contained in a subspace y if $x \leq y$.

For a point a and a subspace x we denote by $\pi(a|x)$ the subspace which contains a and is parallel to x . Axiom (A2) guarantees that there is exactly one such subspace. For every $x \in Q$ we observe that $\theta(x) := \{(a, b) \in A^2 \mid \pi(a|x) = \pi(b|x)\}$ is an equivalence relation on the set of points. We collect a few properties of ordered sets with parallelism which will be useful later.

Proposition 1. *Let $b \in A$ and $x \in Q$. Then*

$$b \leq \pi(a|x) \Leftrightarrow \pi(a|x) = \pi(b|x).$$

Proof. “ \Rightarrow ”: Let $b \leq \pi(a|x)$. By definition $\pi(a|x) \parallel \pi(b|x)$. Also by definition $b \leq \pi(b|x)$. With (A2) it follows that $\pi(a|x) = \pi(b|x)$.

“ \Leftarrow ”: Let $\pi(a|x) = \pi(b|x)$. Since $b \leq \pi(b|x)$, $b \leq \pi(a|x)$ follows immediately. \square

Proposition 2. *Let $a \in A$ and $x \in Q$. Then $[a]\theta(x) = A(\pi(a|x))$.*

Proof. If $b \in [a]\theta(x)$, then $\pi(b|x) = \pi(a|x)$. The previous proposition yields $b \leq \pi(a|x)$, i.e. $b \in A(\pi(a|x))$. The other direction is proved analogously. \square

Proposition 3. *Let $a, b \in A$. Then $a \parallel b$ and $\theta(a) = \Delta_A$ for all $a \in A$.*

Proof. Let $a, b \in A$. It suffices to prove that $\pi(a|b) = a$ as the claims will then follow immediately from the fact that $\pi(a|x) \parallel x$ and from the definition of $\theta(a)$. From the definitions of $\pi(a|b)$ and $\pi(b|a)$ we get $\pi(a|b) \parallel b \leq \pi(b|a) \parallel a$. As $a \leq \pi(a|b)$, Axiom (A3) yields $\pi(a|b) \leq a$. Since $a \leq \pi(a|b)$, by definition, we have $\pi(a|b) = a$, as required. \square

To relate ordered sets with parallelism to each other we introduce homomorphisms:

Definition 7 (homomorphism for ordered sets with parallelism). *For ordered sets with parallelism $\mathbb{A} = (Q, \leq, \parallel)$ and $\mathbb{A}_0 = (Q_0, \leq_0, \parallel_0)$ we call a mapping $\alpha : Q \rightarrow Q_0$ a homomorphism if*

- α maps points to points,
- α is order preserving (i.e. $x \leq y \Rightarrow \alpha(x) \leq_0 \alpha(y)$),
- α preserves the parallelism (i.e. $x \parallel y \Rightarrow \alpha(x) \parallel_0 \alpha(y)$).

By $\text{Hom}(\mathbb{A}, \mathbb{A}_0)$ we denote the set of all homomorphisms from \mathbb{A} to \mathbb{A}_0 .

We will use ordered sets with parallelism to link data tables to affine geometry. To establish a connection to a projective view we provide the concept of a weakly projective ordered set:

Definition 8 (weakly projective ordered set). A triple $\mathbb{P} := (L, \leq, K)$ is called projective ordered set if (L, \leq) is an ordered set (its elements are called objects), $K \subseteq L$ is a set (its elements are called infinity objects) and the axioms (P1) - (P5) hold. The elements of $L \setminus K$ are called affine objects. The set of affine points is defined as $A := \text{Min}(L \setminus K, \leq)$ and we say x is contained in y or y contains x if $x \leq y$.

- (P1) An element of L is an affine object if and only if it contains an affine point.
- (P2) Every affine object x contains a greatest infinity object x_∞ .
- (P3) Every affine point p and every infinity object k are contained in a least element, denoted by $p \vee k$.
- (P4) For every affine point p and every infinity object k we have $(p \vee k)_\infty = k$.
- (P5) For every affine point p contained in an affine object x we have $p \vee x_\infty = x$.

We observe a few properties of a weakly projective ordered set $\mathbb{P} = (L, \leq, K)$:

Proposition 4. Let a and b be affine points. Then $a_\infty = b_\infty$.

Proof. By (P2), a_∞ and b_∞ exist. If $b_\infty \leq a$, that would imply that a_∞ and b_∞ are comparable, because of (P2). Thus $a_\infty = b_\infty$. We assume that a and b_∞ are not comparable. Using (P3), we can form $x = a \vee b_\infty$. Axiom (P4) guarantees, that $b_\infty = (a \vee b_\infty)_\infty = x_\infty$. Clearly, $a_\infty \leq x$ and $b_\infty \leq x$. Because every affine object contains a greatest infinity object and $x_\infty = b_\infty$ we have $a_\infty \leq b_\infty$. This implies again $a_\infty = b_\infty$. \square

Proposition 5. Let x and y be affine objects. Then $x < y$ implies $x_\infty < y_\infty$ and $x \not\leq y_\infty$ (refer to Fig. 1).

Proof. By transitivity $x_\infty < y$. Let a be an affine point contained by both x and y . By (P5) we get $a \vee x_\infty = x$ and $a \vee y_\infty = y$. Therefore $x_\infty \neq y_\infty$. Axiom (P2) requires $x_\infty \leq y_\infty$. This yields $x_\infty < y_\infty$. Now assume $x \leq y_\infty$. This would imply that $a \vee y_\infty = y_\infty$, but this contradicts axiom (P3) which requires $a \vee y_\infty$ to be an affine object. \square

We give a slightly more general statement without proof.

Proposition 6. Let x and y be affine objects. Then $x \leq y$ implies $x_\infty \leq y_\infty$.

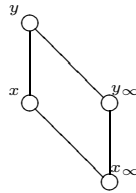


Fig. 1. The only possible configuration for two affine objects and their infinity objects

Proposition 7. *Let a be an arbitrary affine point. Then $\perp := a_\infty$ is the least element of (L, \leq) .*

Proof. Proposition 4 yields that \perp is well defined. Axiom (P1) guarantees that for every affine object x there is an affine point b with $b \leq x$. Using Proposition 6 we deduce $\perp = b_\infty \leq x_\infty$. So \perp is contained by every infinity object and also (by transitivity) contained by every affine object. \square

Again we need a notion of a homomorphism between weakly projective ordered sets:

Definition 9 (homomorphism for weakly projective ordered sets). *For weakly projective ordered sets $\mathbb{P} = (L, \leq, K)$ and $\mathbb{P}_0 = (L_0, \leq_0, K_0)$ we call a mapping $\alpha : L \rightarrow L_0$ a homomorphism if*

- α preserves affine points,
- α preserves affine objects,
- α is order-preserving, and
- α preserves the ∞ -operation, i.e. $\alpha(x_\infty) = \alpha(x)_\infty$.

By $\text{Hom}(\mathbb{P}, \mathbb{P}_0)$ we denote the set of all homomorphisms from \mathbb{P} to \mathbb{P}_0 .

Note that $\alpha(x_\infty) = \alpha(x)_\infty$ implies that infinity objects are mapped to infinity objects by a homomorphism. In the following we will show the categorical equivalence of ordered sets with parallelism and weakly projective ordered sets.

3.1 From Ordered Sets with Parallelism to Weakly Projective Ordered Sets

First from a given ordered set with parallelism $\mathbb{A} = (Q, \leq, \parallel)$ we construct a weakly projective ordered set $\mathbf{P}(\mathbb{A}) := (L_{\mathbb{A}}, \leq_{\mathbb{A}}, K_{\mathbb{A}})$. As affine objects of the weakly projective ordered set we take Q . To construct the necessary infinity objects we make use of the natural equivalence relations $\{\theta(x) \mid x \in Q\}$ on the set of all points of \mathbb{A} . Let $K_{\mathbb{A}} := \{\theta(x) \mid x \in Q\}$ and $L_{\mathbb{A}} := Q \cup K_{\mathbb{A}}$. We define the order

$$\leq_{\mathbb{A}} := \leq \cup \{(\theta(x), y) \mid \exists y' \in Q : x \parallel y' \leq y\} \cup \{(\theta(x), \theta(y)) \mid x \leq y\}.$$

We will show that $(L_{\mathbb{A}}, \leq_{\mathbb{A}}, K_{\mathbb{A}})$ is a weakly projective ordered set. Note that the set of affine points in $\mathbf{P}(\mathbb{A})$ equals the set of points of \mathbb{A} . First, we convince ourselves that the relation $\leq_{\mathbb{A}}$ is an order. The only problem is transitivity. Assume $\theta(x) \leq_{\mathbb{A}} \theta(y) \leq_{\mathbb{A}} z$ where $x, y, z \in Q$. We want to show that $\theta(x) \leq_{\mathbb{A}} z$ which is equivalent to $\exists x' \in Q : x \parallel x' \leq z$. Since $\theta(y) \leq_{\mathbb{A}} z$ there exists y' with $y \parallel y' \leq z$. Also we have $x \leq y$. Let $a \in A(y')$. We get $\pi(a|x) \parallel x \leq y \parallel y'$ with $A(\pi(a|x)) \cap A(y') \neq \emptyset$. With (A3) it follows that $\pi(a|x) \leq y'$. But by transitivity of \leq we get $x \parallel \pi(a|x) \leq z$. This implies $\theta(x) \leq_{\mathbb{A}} z$. Now we show (P1)-(P5). Take an arbitrary affine object $k \in Q$. Axiom (A1) guarantees that there exists a point below k ; this shows (P1). For verifying (P2) consider an affine object x .

The infinity object $\theta(x)$ is contained by x by definition of $\leq_{\mathbb{A}}$. Let $\theta(y)$ be another infinity object contained by x . By definition of $\leq_{\mathbb{A}}$ this implies $x \parallel y$. But this implies $\theta(x) = \theta(y)$. Thus $\theta(x)$ is the unique greatest infinity object contained by x . To validate (P3) we consider an affine point p together with an infinity object $\theta(x)$. The affine object $\pi(p|x)$ contains p and also – by definition of $\leq_{\mathbb{A}}$ – contains $\theta(x)$. Any other affine object covering $\theta(x)$ has to be parallel to x and hence can not contain p . Using our previous results, a simple computation shows (P4): $(p \vee k)_{\infty} = (p \vee \theta(x))_{\infty} = (\pi(p|x))_{\infty} = \theta(x)$. For (P5) let p be an affine point and x be an affine object with $p \leq x$. Then $p \vee x_{\infty} = p \vee \theta(x) = \pi(p|x) = x$. This concludes our argument that $\mathbf{P}(\mathbb{A})$ is a weakly projective ordered set.

Now we extend \mathbf{A} to $\text{Hom}(\mathbb{A}, \mathbb{A}_0)$. Let $\alpha \in \text{Hom}(\mathbb{A}, \mathbb{A}_0)$. We will define the mapping $\mathbf{P}(\alpha) : \mathbf{P}(\mathbb{A}) \rightarrow \mathbf{P}(\mathbb{A}_0)$ and show in the following that it is indeed a homomorphism between the corresponding weakly projective ordered sets. Set $\mathbf{P}(\alpha)(x) := \alpha(x)$ for $x \in L \setminus K$ and $\mathbf{P}(\alpha)(\theta(x)) := \theta(\alpha(y))$ for $\theta(x) \in K$ and $\theta(x) \prec y$, with y an affine object (we call y an *affine cover* of $\theta(x)$). Obviously, the mapping $\mathbf{P}(\alpha)$ is well-defined for $x \in L \setminus K$. For $\theta(x) \in K$ all the affine covers of $\theta(x)$ are parallel in \mathbb{A} . Thus all the images of these covers are parallel in \mathbb{A}_0 , because α is a homomorphism. Hence, it does not matter which affine cover y of $\theta(x)$ we use to determine $\theta(\alpha(y))$, because all the images of covers are parallel and yield the same infinity object in $\mathbf{P}(\mathbb{A}_0)$.

Clearly, $\mathbf{P}(\alpha)$ preserves affine points, affine objects and infinity objects. We have to show that $\mathbf{P}(\alpha)$ is order-preserving. Let $x \leq_{\mathbb{A}} y$. If both, x and y , are affine objects in $\mathbf{P}(\mathbb{A})$ it follows immediately that $\mathbf{P}(\alpha)(x) \leq_{Q_0} \mathbf{P}(\alpha)(y)$. If both, x and y , are infinity objects they are of the form $\theta(x')$ and $\theta(y')$ for affine objects $x' \prec_{\mathbb{A}} x'$ and $y' \prec_{\mathbb{A}} y'$ with $x' \leq_{\mathbb{A}} y'$. That implies $x' \leq y'$ because x' and y' are affine objects. Therefore $\alpha(x') \leq \alpha(y')$. This implies (by definition of $\leq_{\mathbb{A}_0}$) $\theta(\alpha(x')) \leq_{\mathbb{A}_0} \theta(\alpha(y'))$. If x is an infinity object and y is an affine object, we can denote x by $\theta(x')$ for some affine object x' . By definition of $\leq_{\mathbb{A}}$, it follows that there exists an affine object y' with $x' \parallel y' \leq y$ in \mathbb{A} . That implies $\alpha(y') \leq_0 \alpha(y)$. Again by definition of $\leq_{\mathbb{A}_0}$ it follows that $\theta(\alpha(y')) \leq_{\mathbb{A}_0} \alpha(y)$. That completes our argument that $\mathbf{P}(\alpha)$ is order-preserving (the last case – x an affine object and y an infinity object – can not occur) and therefore a homomorphism.

3.2 From Weakly Projective Ordered Sets to Ordered Sets with Parallelism

For the reverse construction we start with a weakly projective ordered set $\mathbb{P} = (L, \leq, K)$ and build an ordered set with parallelism $\mathbf{A}(\mathbb{P}) := (Q_{\mathbb{P}}, \leq_{\mathbb{P}}, \parallel_{\mathbb{P}})$. We take the set of affine objects of the weakly projective ordered set as underlying set of our ordered set with parallelism $Q_{\mathbb{P}} := L \setminus K$. The order on $Q_{\mathbb{P}}$ is the restriction of the order \leq to $Q_{\mathbb{P}}$ (i.e. $\leq_{\mathbb{P}} := \leq \cap (L \setminus K)^2$). The parallelism is derived via the infinity objects: $\parallel_{\mathbb{P}} := \{(x, y) \in Q_{\mathbb{P}}^2 \mid x_{\infty} = y_{\infty}\}$. Obviously, $\parallel_{\mathbb{P}}$ is an equivalence relation. We have to show that (A1) - (A3) hold in $\mathbf{A}(\mathbb{P})$. Axiom (P1) translates directly into (A1). For (A2) consider a minimal element $a \in Q_{\mathbb{P}}$ and an arbitrary element $x \in Q_{\mathbb{P}}$. With (P2), x contains a greatest infinity object x_{∞} in L . Axiom (P3) allows to form $y = a \vee x_{\infty}$. Clearly, $a \leq_{\mathbb{P}} y$. We

need to show that $y \parallel_{\mathbb{P}} x$ which is equivalent to $y_{\infty} = x_{\infty}$. Using (P4), we see that $y_{\infty} = (a \vee x_{\infty})_{\infty} = x_{\infty}$. To verify (A3), we assume $x' \parallel_{\mathbb{P}} x \leq_{\mathbb{P}} y \parallel_{\mathbb{P}} y'$ and $A(x') \cap A(y') \neq \emptyset$ for affine objects x, x', y, y' . Proposition 6 yields $x_{\infty} \leq_{\mathbb{P}} y_{\infty}$. The parallelity gives $x_{\infty} = y_{\infty}$. The inequality $A(x') \cap A(y') \neq \emptyset$ ensures that there exists an affine point a contained by both x' and y' . Axiom (P3) allows us to form $a \vee x'_{\infty}$ which is equal to x' by (P5). Clearly, $a \vee y'_{\infty} = y'$ is also an upper bound of a and x_{∞} . Because there has to be a least one, $x' \leq_{\mathbb{P}} y'$. This finishes our argument that $\mathbf{A}(\mathbb{P})$ is an ordered set with parallelism.

Now we extend \mathbf{A} to $\text{Hom}(\mathbb{P}, \mathbb{P}_0)$. Let $\alpha \in \text{Hom}(\mathbb{P}, \mathbb{P}_0)$. We will define the mapping $\mathbf{A}(\alpha) : \mathbf{A}(\mathbb{P}) \rightarrow \mathbf{A}(\mathbb{P}_0)$ and show in the following that it is indeed a homomorphism between the corresponding ordered sets with parallelism. We set $\mathbf{A}(\alpha)(x) := \alpha|_{L \setminus K}(x)$. Clearly, $\mathbf{A}(\alpha)$ preserves points and is order-preserving. Let $x \parallel_{\mathbb{P}} y$. This translates into $x_{\infty} = y_{\infty}$ in \mathbb{P} . Because α is a homomorphism it holds that $\alpha(x)_{\infty} = \alpha(x_{\infty})$. This implies $\alpha(x)_{\infty} = \alpha(y)_{\infty}$. We get $x \parallel_{\mathbb{P}_0} y$. Therefore $\mathbf{A}(\alpha)$ is a homomorphism.

3.3 Categorical Equivalence

Having defined the functors \mathbf{P} and \mathbf{A} we will describe the categorical equivalence between ordered sets with parallelism and weakly projective ordered sets. To keep this part short we will sketch only the main ideas for the proofs (using previously explicated notation implicitly).

For each ordered set with parallelism \mathbb{A} , the map $\epsilon_{\mathbb{A}} := \text{id}_Q$ is an isomorphism from \mathbb{A} to $\mathbf{A}(\mathbf{P}(\mathbb{A}))$. For each weakly projective ordered set \mathbb{P} , an isomorphism $\epsilon_{\mathbb{P}} : \mathbb{P} \rightarrow \mathbf{P}(\mathbf{A}(\mathbb{P}))$ is defined via $\epsilon_{\mathbb{P}}|_{L \setminus K} := \text{id}_{L \setminus K}$ and $\epsilon_{\mathbb{P}}(k) := \theta(x)$ where $k \in K$ and x is an affine cover of k .

Proposition 8. *For ordered sets with parallelism \mathbb{A} and \mathbb{A}_0 the mapping \mathbf{AP} is a bijection between $\text{Hom}(\mathbb{A}, \mathbb{A}_0)$ and $\text{Hom}(\mathbf{A}(\mathbf{P}(\mathbb{A})), \mathbf{A}(\mathbf{P}(\mathbb{A}_0)))$ which makes the following diagram commutative for every $\alpha \in \text{Hom}(\mathbb{A}, \mathbb{A}_0)$:*

$$\begin{array}{ccc} \mathbb{A} & \xrightarrow{\alpha} & \mathbb{A}_0 \\ \epsilon_{\mathbb{A}} \downarrow & & \downarrow \epsilon_{\mathbb{A}_0} \\ \mathbf{AP}(\mathbb{A}) & \xrightarrow{\mathbf{AP}(\alpha)} & \mathbf{AP}(\mathbb{A}_0) \end{array}$$

Proof. Let $x \in Q$. We will show that the diagram is commutative:

$\mathbf{AP}(\alpha)(\epsilon_{\mathbb{A}}(x)) = \mathbf{AP}(\alpha)(x) = (\mathbf{P}(\alpha))|_{L \setminus K}(x) = \alpha(x)$ and $\epsilon_{\mathbb{A}_0}(\alpha(x)) = \alpha(x)$. Also, the first equation expresses that $\mathbf{AP} = \text{id}_{\text{Hom}(\mathbb{A})}$. \square

Proposition 9. *For weakly projective ordered sets \mathbb{P} and \mathbb{P}_0 the mapping \mathbf{PA} is a bijection between $\text{Hom}(\mathbb{P}, \mathbb{P}_0)$ and $\text{Hom}(\mathbf{P}(\mathbf{A}(\mathbb{P})), \mathbf{P}(\mathbf{A}(\mathbb{P}_0)))$ which makes the following diagram commutative for every $\alpha \in \text{Hom}(\mathbb{P}, \mathbb{P}_0)$:*

$$\begin{array}{ccc}
\mathbb{P} & \xrightarrow{\alpha} & \mathbb{P}_0 \\
\epsilon_{\mathbb{P}} \downarrow & & \downarrow \epsilon_{\mathbb{P}_0} \\
\mathbf{PA}(\mathbb{P}) & \xrightarrow{\mathbf{PA}(\alpha)} & \mathbf{PA}(\mathbb{P}_0)
\end{array}$$

Proof. We will show that the diagram is commutative. For $x \in L \setminus K$ we have: $\mathbf{PA}(\alpha)(\epsilon_{\mathbb{P}}(x)) = \mathbf{PA}(\alpha)(x) = (\mathbf{P}(\alpha|_{L \setminus K}))(x) = \alpha(x)$ and $\epsilon_{\mathbb{P}_0}(\alpha(x)) = \alpha(x)$. Let $x \in K$ and y be an affine cover of x .

$$\mathbf{PA}(\alpha)(\epsilon_{\mathbb{P}}(x)) = \mathbf{PA}(\alpha)(\theta(y)) = \theta(\mathbf{A}(\alpha)(y)) = \theta(\alpha(y))$$

and

$$\epsilon_{\mathbb{P}_0}(\alpha(x)) = \theta(y') \text{ with } y' \text{ affine cover of } \alpha(x).$$

Since y is an affine cover of x , $\alpha(y)$ is an affine cover of $\alpha(x)$. This guarantees $\theta(y') = \theta(\alpha(y))$. It is easy to see that \mathbf{PA} is bijective. \square

The previous propositions lead to the following result:

Theorem 1. *The categories of ordered sets with parallelism and weakly projective ordered sets are equivalent.* \square

4 Systems of Equivalence Relations and Affine Ordered Sets

To be able to establish a close connection between SERs and ordered sets with parallelism we have to look at a certain subclass of ordered sets with parallelism.

Definition 10 (affine ordered set). *An affine ordered set is an ordered set with parallelism where the following additional axiom holds (for notation compare definition 6):*

$$\begin{aligned}
(\mathbf{A4}) \quad & \forall x, y \in Q \exists x', y' \in Q : x \not\leq y \ \& \ A(x) \subseteq A(y) \\
& \Rightarrow x' \parallel x \ \& \ y' \parallel y \ \& \ A(x') \cap A(y') \neq \emptyset \ \& \ A(x') \not\subseteq A(y').
\end{aligned}$$

The next proposition explains this axiom.

Proposition 10. *Let $\mathbb{A} = (Q, \leq, \parallel)$ be an affine ordered set and $x, y \in Q$. Then $x \leq y$ is equivalent to $\theta(x) \subseteq \theta(y)$ & $A(x) \subseteq A(y)$.*

Proof. “ \Rightarrow ”: Suppose $x \leq y$. If $(a, b) \in \theta(x)$ there exists $x' \parallel x$ with $a, b \leq x'$. By (A2) there exists a unique $y' \parallel y$ with $a \leq y'$. By (A3) $x' \leq y'$, hence $(a, b) \in \theta(y)$. Obviously, $A(x) \subseteq A(y)$.

“ \Leftarrow ”: Suppose $\theta(x) \subseteq \theta(y)$ with $A(x) \subseteq A(y)$. Suppose $x \not\leq y$. Axiom (A4) yields the existence of x' and y' with $x' \parallel x$ and $y' \parallel y$ where $A(x) \cap A(y) \neq \emptyset$. Let $a \in A(x') \cap A(y')$. Also (A4) yields the existence of a point $b \in A(x')$ with $b \notin A(y')$. Therefore $(a, b) \in \theta(x)$ but $(a, b) \notin \theta(y)$, a contradiction.

We will show that the concept of an affine ordered set captures the essence of a SER with identity relation and vice versa.

4.1 Moving between Affine Ordered Sets and SERs with Identity Relation

For constructing the SER with identity relation corresponding to a given ordered set with parallelism $\mathbb{A} = (Q, \leq, \parallel)$ we take the set of all points A of the ordered set and use it as carrier set for the SER. The equivalence relations are recovered using the parallelism \parallel . We know from the last section that $\theta(x) := \{(a, b) \in A^2 \mid \pi(a|x) = \pi(b|x)\}$ is an equivalence relation for every $x \in Q$. Therefore we can define $\mathbf{E}(\mathbb{A}) := (\text{Min}(Q, \leq), \{\theta(x) \mid x \in Q\})$. Proposition 3 yields that for any point a the corresponding relation $\theta(a)$ equals Δ_A . Hence $\Delta_A \in \{\theta(x) \mid x \in Q\}$ and $\mathbf{E}(\mathbb{A})$ is a SER with identity relation.

For describing the reverse construction let $\mathbb{E} = (D, E)$ be a SER with identity relation. We consider the set of all labelled equivalence classes $Q := \{([x]\theta, \theta) \mid x \in D, \theta \in E\}$ as set of subspaces of an ordered set with parallelism. We define the order on Q as

$$([x]\theta_1, \theta_1) \leq' ([y]\theta_2, \theta_2) : \iff [x]\theta_1 \subseteq [y]\theta_2 \wedge \theta_1 \subseteq \theta_2.$$

Note that, because $\Delta_D \in E$, we have

$$\text{Min}(\{([x]\theta, \theta) \mid \theta \in E\}, \leq') = \{(\{x\}, \Delta_D) \mid x \in D\}.$$

That means the points of the constructed ordered set can be identified with the carrier set of the SER.

To complete the construction we define a relation \parallel' on the set of equivalence classes:

$$([x]\theta_1, \theta_1) \parallel' ([y]\theta_2, \theta_2) : \iff \theta_1 = \theta_2.$$

We have to show that $\mathbf{A}(\mathbb{E}) := (\{([x]\theta, \theta) \mid \theta \in E\}, \leq', \parallel')$ is indeed an ordered set with parallelism.

Obviously (A1) holds. For verifying (A2), consider an arbitrary subspace $x = ([b]\theta, \theta)$ and a point $a = ([a]\theta_0, \theta_0)$. We set $t := ([a]\theta, \theta)$. Then t is the unique subspace t with $a \leq' t \parallel' x$. For showing (A3), let $(X, \theta_1), (X', \theta_1), (Y, \theta_2)$ and (Y', θ_2) be subspaces of $\mathbf{A}(\mathbb{E})$ with $(X, \theta_1) \leq' (Y, \theta_2)$ and $A((X', \theta_1)) \cap A((Y', \theta_2)) \neq \emptyset$. That means $X \subseteq Y$, $\theta_1 \subseteq \theta_2$ and $X' \cap Y' \neq \emptyset$. This implies $X' \subseteq Y'$ because $\theta_1 \subseteq \theta_2$. This concludes the argument that $\mathbf{A}(\mathbb{E})$ is an ordered set with parallelism.

To prove that this ordered set with parallelism is an affine ordered set we validate (A4). Let $x = ([a]\theta_1, \theta_1)$ and $y = ([b]\theta_2, \theta_2)$ be arbitrary subspaces with $x \not\leq' y$ and $A(x) \subseteq A(y)$. Since $x \not\leq' y$, it follows that either $[a]\theta_1 \not\subseteq [b]\theta_2$ or $\theta_1 \not\subseteq \theta_2$. If $[a]\theta_1 \not\subseteq [b]\theta_2$, then there exists an atom c in $A(x)$ with $c \notin A(y)$. Let $x' := x$ and $y' := ([c]\theta_2, \theta_2)$. Obviously $x \parallel x'$ and $y \parallel y'$. Because $c \in A(x')$ and $c \in A(y')$, we have $A(x') \cap A(y') \neq \emptyset$. With $A(x') = A(x) \subseteq A(y)$ we get $A(x') \not\subseteq A(y')$ using that $A(y')$ and $A(y)$ are disjoint. If $\theta_1 \not\subseteq \theta_2$, there exists an equivalence class $[c]\theta_1$ with $[c]\theta_1 \supset [c]\theta_2$. We can choose $x' := [c]\theta_1$ and $y' := [c]\theta_2$. Obviously the conclusion is fulfilled. Therefore $\mathbf{A}(\mathbb{E})$ is even an affine ordered set.

4.2 Categorical Equivalence between Affine Ordered Sets and SERs with Identity Relation

First we start with an affine ordered set $\mathbb{A} := (Q, \leq, \parallel)$. We propose that $\mathbf{AE}(\mathbb{A}) \simeq \mathbb{A}$. Because \mathbf{E} transfers the points of \mathbb{A} into the carrier set of the SER, the points of the resulting affine ordered set are exactly the singletons of its former points labelled with the identity relation. More general, every $x \in Q$ is transformed into the set of points it contains, i.e. $A(x)$, together with a label. Let $\mathbf{EA}(\mathbb{A}) := (Q_0, \leq_0, \parallel_0)$. Then the idea is to define a bijection $\alpha : Q \rightarrow Q_0$ where $\alpha(x) := (A(x), \theta(x))$. We have to ensure that the order and the parallelism are preserved. Let $x \leq y$. By Proposition 10, this implies $A(x) \subseteq A(y)$ and $\theta(x) \subseteq \theta(y)$ which means exactly $\alpha(x) \leq_0 \alpha(y)$. Now suppose $x \parallel x'$. We get $\alpha(x) = (A(x), \theta(x)) \parallel_0 (A(x'), \theta(x')) = \alpha(x')$ because $\theta(x) = \theta(x')$. Hence α is an isomorphism.

Let $\mathbb{E} = (D, E)$ be a SER with identity relation. We show that $\mathbb{E} \simeq \mathbf{EA}(\mathbb{E})$. The points of the affine ordered set $\mathbf{A}(\mathbb{E})$ can be identified with the singletons of D . Let $\mathbf{EA}(\mathbb{E}) := (D_0, E_0)$. Then we can define bijections $\alpha : D \rightarrow D_0$ where $\alpha(d) := (\{d\}, \Delta_D)$ and $\beta : E \rightarrow E_0$ where $\beta(\theta') := \{((a, \Delta_D), (b, \Delta_D)) \mid (a, b) \in \theta'\}$. Obviously, (α, β) is an isomorphism between \mathbb{E} and $\mathbf{EA}(\mathbb{E})$.

Because it is easy to see that results analogous to those in Sect.3.3 can be formulated for SERs with identity relation and affine ordered sets, we omit the detailed description and give only one additional definition specializing weakly projective ordered sets before our main results are presented in the next section.

Definition 11 (projective ordered sets). *A projective ordered set is a weakly projective ordered set where the following additional axiom holds (for notation compare definition 8):*

(P6) *If an infinity object k is not contained in an infinity object l , then there are affine points a and b with $a \vee k = b \vee k$ and $a \vee l \neq b \vee l$.*

5 Main Results

We combine our main results in an equivalence theorem for four different categories.

Main Theorem 2. *The following four categories are equivalent:*

- *category of simple many-valued contexts with id-attribute*
- *category of systems of equivalence relations with identity relation*
- *category of affine ordered sets*
- *category of projective ordered sets*

Proof. Only the categorical equivalence between affine and projective ordered sets remains to be proven. Using the categorical equivalence between ordered sets with parallelism and weakly projective ordered sets (as shown in section 3) the proof narrows down to showing that the properties (A4) and (P6) are equivalent.

“(A4) \implies (P6)”: Let \mathbb{A} be an affine ordered set and $\mathbf{P}(\mathbb{A})$ the corresponding weakly projective ordered set. We show that $\mathbf{P}(\mathbb{A})$ is a projective ordered set. Let $k = \theta(x)$ and $l = \theta(y)$ be infinity objects with $k \not\leq l$ and a be an affine point. If $A(a \vee \theta(x)) \not\subseteq A(a \vee \theta(y))$, let $b \in A(a \vee \theta(x)) \setminus A(a \vee \theta(y))$. Then $a \vee \theta(x) = b \vee \theta(x)$ and $a \vee \theta(y) \neq b \vee \theta(y)$. If $A(a \vee \theta(x)) \subseteq A(a \vee \theta(y))$ we can apply (A4) which yields affine objects x' and y' with $x'_\infty = \theta(x)$, $y'_\infty = \theta(y)$, $A(x') \cap A(y') \neq \emptyset$, and $A(x') \not\subseteq A(y')$. Therefore we can pick affine points b and c with $b \in A(x') \cap A(y')$ and $c \in A(x') \setminus A(y')$. Again we get $b \vee \theta(x) = x' = c \vee \theta(x)$ and $b \vee \theta(y) = y' \neq c \vee \theta(y)$. This shows that (P6) holds in $\mathbf{P}(\mathbb{A})$.

“(P6) \implies (A4)”: Let \mathbb{P} be an projective ordered set and $\mathbf{A}(\mathbb{P})$ the corresponding ordered set with parallelism. We show that $\mathbf{A}(\mathbb{P})$ is an affine ordered set. Let x and y be affine objects with $x \not\leq y$ and $A(x) \subseteq A(y)$. This implies that $x_\infty \not\leq y_\infty$. Using (P6) we deduce the existence of affine points a and b with $a \vee x_\infty = b \vee x_\infty$ and $a \vee y_\infty \neq b \vee y_\infty$. Let $x' := \pi(a|x)$ and $y' := \pi(a|y)$. Then $x' \parallel x$ and $y' \parallel y$, $a \in A(x') \cap A(y')$, and $b \in A(x') \setminus A(y')$. Therefore (A4) holds in $\mathbf{A}(\mathbb{P})$. \square

We conclude our investigation with an example which activates the main theorem and shows a possible application in data analysis and data visualization.

6 An Example

Suppose we want to buy a car. We might investigate a few models and record their properties. The result can look like a sample data table from [Pa91] as shown in Fig.2.

Car	Prize	Mileage	Size	Max-Speed	Acceleration
1	low	medium	full	low	good
2	medium	medium	compact	high	poor
3	high	medium	full	low	good
4	medium	medium	full	high	excellent
5	low	high	full	low	good

Fig. 2. Car context

Now, our interest could be to determine dependencies between the attributes or on a finer level even between attribute values.

Applying our equivalence theorem we are able to translate the data table – which can be seen as a many-valued context – into a SER with identity relation where $G = \{1, 2, 3, 4, 5\}$ and the set of equivalence relations is given by the following partitions:

$$\begin{aligned}
 P_{\text{Prize}} &= \{\{1, 5\}, \{2, 4\}, \{3\}\}, & P_{\text{Mileage}} &= \{\{1, 2, 3, 4\}, \{5\}\}, \\
 P_{\text{Size}} &= \{\{1, 3, 4, 5\}, \{2\}\}, & P_{\text{Max-Speed}} &= \{\{1, 3, 5\}, \{2, 4\}\}, \\
 P_{\text{Acceleration}} &= \{\{1, 3, 5\}, \{2\}, \{4\}\}, & \text{and } \Delta_G &.
 \end{aligned}$$

The Hasse-diagram in Fig.3 visualizes the set-theoretic inclusion order between the equivalence relations.

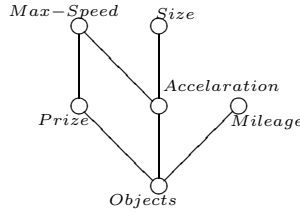


Fig. 3. Order of the equivalence relations

From this diagram we can read off the following functional dependencies between attributes:

$\text{Prize} \Rightarrow \text{Max-Speed}$; $\text{Acceleration} \Rightarrow \text{Max-Speed}$; $\text{Acceleration} \Rightarrow \text{Size}$.

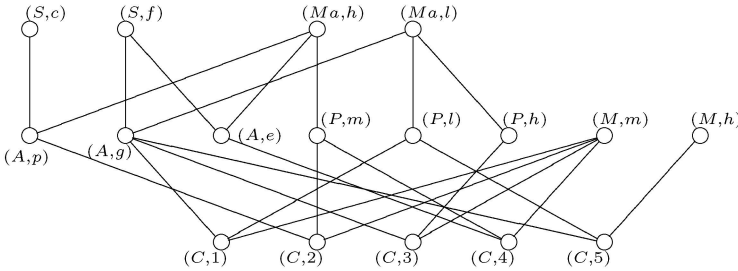


Fig. 4. Affine ordered set for the car context (S=Size, Ma=Max-Speed, A=Acceleration, P=Prize, M=Mileage; c=compact, f=full, h=high, m=medium, l=low, p=poor, e=excellent)

We can construct an affine ordered set from this SER with identity relation to get an order-theoretic representation of our data table as shown in Fig.4. Each node represents an equivalence class with regard to one attribute. Therefore it is labelled with a pair consisting of the attribute and the value which describes the represented class. If we follow downward paths from an equivalence class we can read off its elements at the bottom level. If we follow upward paths from a node we can read off value dependencies like “(Acceleration,poor) implies (Size,compact)”. There are many options for improving the readability and expressiveness of the diagram, but this is not in the scope of our article. A Hasse-diagram for the corresponding projective ordered set is shown in Fig.5.

It was demonstrated that the connection of order-theory and geometry can yield useful instruments for data visualization and classification. Our diagrammatic representations of data make it easier for humans to detect structural patterns and interesting anomalies. Importantly, these representations can be translated back to the original data table and vice versa without losing information.

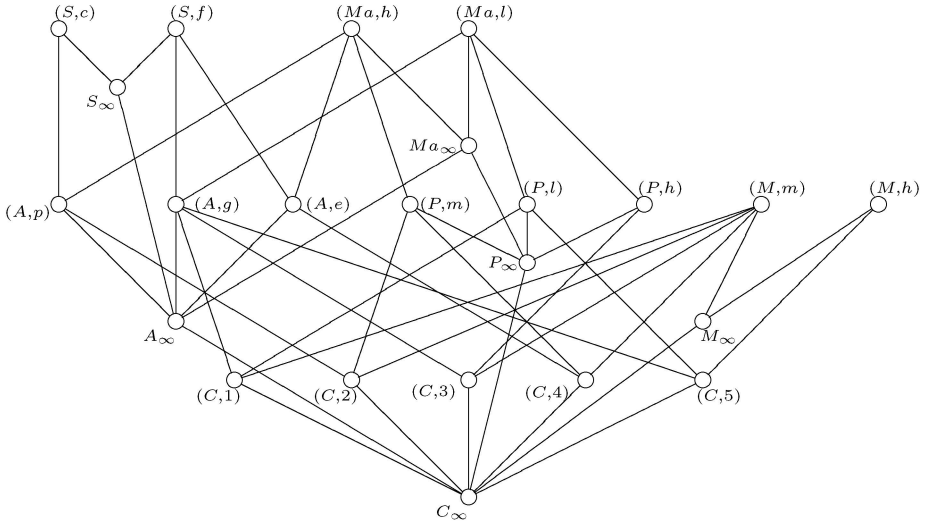


Fig. 5. Projective ordered set for the car context

References

- [GW99] B. Ganter, R. Wille: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin – Heidelberg – New York (1999)
- [Pa91] Z. Pawlak: Rough Sets - Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht – Boston – London (1991)
- [Pr95] V.R. Pratt: Chu spaces and their interpretation as concurrent objects. In J. van Leeuwen, editor, Computer Science Today: Recent Trends and Developments, volume 1000 of LNCS, pages 392–405. Springer, Berlin – Heidelberg – New York (1995)
- [Wi70] R. Wille: Kongruenzklassengeometrien. Springer, Berlin – Heidelberg – New York (1970)

Concept Extensions and Weak Clusters Associated with Multiway Dissimilarity Measures

Jean Diatta

IREMIA, Université de la Réunion
15 avenue René Cassin - BP 7151
97715 Saint-Denis messag cedex 9, France
Jean.Diatta@univ-reunion.fr

Abstract. We consider contexts with a finite set of entities described in a poset. When entity descriptions belong to a meet-semilattice, we show that nonempty extensions of concepts assigned to such a context coincide with weak clusters associated with pairwise or multiway dissimilarity measures satisfying some compatibility condition. Moreover, by duality principle, when entity descriptions belong to a join-semilattice, a similar result holds for so-called dual concepts of the given context.

1 Introduction

When a dataset to be treated is not presented in the form of a formal context, a current formal concept analysis (FCA) approach is to bring it into this form by a process called “scaling”. Another approach consists in generalizing the construction of concepts to contexts where the entity set or the attribute set is considered as a poset [1,2,3,4,5]. The present paper belongs to the framework of this approach. Indeed, we are concerned with concepts assigned to contexts where the entity description space is a poset. When the entities are in finite number and their description space a meet-semilattice, we show that a nonempty entity subset is a concept extension if and only if it is a weak cluster associated with some pairwise or multiway dissimilarity measure satisfying some compatibility condition. Moreover, by duality principle, when the entity description space is a join-semilattice, a similar result holds for so-called dual concepts. It should be noticed that weak clusters associated with a (multiway) dissimilarity measure are nonempty subsets with a positive weak isolation degree [6,7]. Therefore, besides the connection established between cluster analysis and FCA, this paper shows on the one hand that to each nonempty concept extension is associated some (multiway) dissimilarity-based isolation degree which, in turn, can be considered as a quantitative concept selection criterion. On the other hand, it reveals a (multiway) dissimilarity type whose associated weak clusters are easily interpretable, as being concept extensions. In addition, the presented result has good potential to be of help for applications where both cluster and concept analysis methods are likely to be useful. The paper is structured in five sections.

Section 2 introduces concepts assigned to contexts where entity descriptions belong to a meet or join-semilattice. Multiway dissimilarity measures and their associated weak clusters are presented in Section 3. The relation between (dual) concept extensions and weak clusters is given in Section 4, and the paper is closed with a brief conclusion.

2 Concepts in Contexts

Where Entity Descriptions Belong to a Poset

It is well known that given a formal context $\mathbb{K} = (E, A, R)$, the binary relation R induces a Galois connection between the posets $(\mathcal{P}(E), \subseteq)$ and $(\mathcal{P}(A), \subseteq)$ by means of the maps

$$f : X \mapsto \bigcap_{a \in X} \{a \in A : (x, a) \in R\}$$

and

$$g : I \mapsto \bigcap_{a \in I} \{x \in E : (x, a) \in R\},$$

for $X \subseteq E$ and $I \subseteq A$ [8,9]. A *formal concept* of \mathbb{K} is a pair $c = (X, I)$ such that $f(X) = I$ and $g(I) = X$.

As the Galois connection (f, g) relates two posets, the construction of formal concepts naturally extends to contexts where elements of E are viewed as described in a poset. Following [4], we denote such a context as a triple $(E, \underline{D}, \delta)$ where E stands for the entity set, $\underline{D} := (D, \leq)$ the entity description space, and δ the descriptor that maps E into \underline{D} . When \underline{D} is a complete meet-semilattice, the descriptor δ induces a Galois connection between $(\mathcal{P}(E), \subseteq)$ and \underline{D} by means of the maps

$$f : X \mapsto \inf \{\delta(x) : x \in X\}$$

and

$$g : \omega \mapsto \{x \in E : \omega \leq \delta(x)\},$$

for $X \subseteq E$ and $\omega \in D$. In these conditions, the map $\phi_\delta := g \circ f$ is a closure operator [10], and a *concept* assigned to $(E, \underline{D}, \delta)$ will be any pair $c = (X, \omega)$ such that $\phi_\delta(X) = X$ and $f(X) = \omega$; the set X will be called the *extension* of c . It may be noted that such concepts are considered in [1,4,5]. When \underline{D} is a complete join-semilattice, the descriptor δ induces a Galois connection between $(\mathcal{P}(E), \subseteq)$ and the order-dual of \underline{D} by means of the maps

$$f^\partial : X \mapsto \sup \{\delta(x) : x \in X\}$$

and

$$g^\partial : \omega \mapsto \{x \in E : \omega \geq \delta(x)\},$$

for $X \subseteq E$ and $\omega \in D$. In these conditions, the map $\phi_\delta^\partial := g^\partial \circ f^\partial$ is a closure operator, and a so-called *dual concept* of $(E, \underline{D}, \delta)$ will be any pair $c = (X, \omega)$ such that $\phi_\delta^\partial(X) = X$ and $f^\partial(X) = \omega$; here again, the set X will be called the *extension* of c . It may also be noted that these dual concepts are the elements of what is called in [2] the union Galois lattice of $(E, \underline{D}, \delta)$.

In what follows, E will denote a finite nonempty entity set, \underline{D} is a meet-semilattice and δ a descriptor that maps E into \underline{D} . For all $X \subseteq E$, $\delta(X)$ will denote the set of descriptions of entities belonging to X .

Consider Table 1 presenting five visitors of a given Web site, described by three attributes: LiLo, NoLi, ReSu, where $\text{LiLo}(x)$ is the login-logout time interval of visitor x within the interval 0-24, $\text{NoLi}(x)$ is the number of times visitor x logs in at $\text{LiLo}(x)$ interval during a given fixed period, and $\text{ReSu}(x)$ is the subjects requested by x during a session; requested subjects are sets of subjects from: Arts & Humanities (AH), Business & Economy (BE), Computers & Internet (CI), News & Media (NM), Recreation & Sports (RS), Science & Health (SH), Society & Culture (SC).

Table 1. Example context \mathbb{K}_1 where entity descriptions belong to a poset

	LiLo	NoLi	ReSu
1	0-2	30	CI,RS
2	21-24	35	AH,NM,SC
3	0-3	40	AH,BE,CI,RS
4	22-24	35	AH,SC
5	12-14	30	BE,NM

Then Table 1 can be seen as representing a context $\mathbb{K}_1 := (E_1, \underline{D}_1, \delta_1)$ where E_1 is the set $\{1, 2, 3, 4, 5\}$, \underline{D}_1 the direct product of three posets: the set $(\text{CI}([0, 24]), \subseteq)$ of finite unions of closed intervals of $[0, 24]$ endowed with set inclusion order, the set $([30; 40], \leq)$ of integers from 30 to 40, endowed with integer usual order, and the powerset $(\mathcal{P}(S), \subseteq)$ of the set $S = \{AH, BE, CI, NM, RS, SC\}$, endowed with set inclusion order, and $\delta_1(x) = (\text{LiLo}(x), \text{NoLi}(x), \text{ReSu}(x))$. The pair $(\{1, 3\}, (0-2, 30, \{CI, RS\}))$ is a concept of \mathbb{K}_1 ; but $\{1, 2, 3\}$ is not a \mathbb{K}_1 concept extension because $\inf \delta_2(\{1, 2, 3\}) = (\emptyset, 30, \emptyset) \leq \delta_2(4)$. On the other hand, the pair $(\{4, 5\}, (12-14 \cup 22-24, 35, \{AH, BE, NM, SC\}))$ is a dual concept of \mathbb{K}_1 , where $12-14 \cup 22-24$ denotes the union of intervals $[12, 14]$ and $[22, 24]$; but $\{1, 2, 3\}$ is not a \mathbb{K}_1 dual concept extension because $\delta_2(4) \leq \sup \delta_2(\{1, 2, 3\}) = (0-3 \cup 21-24, 40, \{AH, BE, CI, NM, RS, SC\})$.

3 Weak Clusters Associated with Multiway Dissimilarity Measures

Multiway dissimilarity measures are natural extensions of classical pairwise dissimilarity measures, that allow global comparison of more than two entities. In the last decade, they have been investigated or considered from different approaches in many works among which we just mention [7, 11, 12].

For all integer $p \geq 1$ and all nonempty subset $X \subseteq E$, $X_{\leq p}^*$ will denote the set of nonempty subsets of X with at most p elements. A *multiway dissimilarity* measure on E will be any nonnegative real valued map d defined on a set of

nonempty subsets of E , such that $d(X) \leq d(Y)$ when $X \subseteq Y$. Sometimes we will be interested in so-called k -way dissimilarity measures, where k is an integer such that $k \geq 2$, i.e., multiway dissimilarity measures defined on $E_{\leq k}^*$. On the other hand, we will simply write $d(x)$ or $d(x, y)$ for $x, y \in E$. Classical pairwise dissimilarity measures correspond to the case $k = 2$. It should also be noticed that the usual condition $d(x) = 0$ is not required in the present paper.

Example 1. Consider the context \mathbb{K}_1 given in Table 1. Then the map dis defined on $\mathcal{P}(E_1) \setminus \{\emptyset\}$ by

$$\text{dis}(X) = 47 - \lambda(\cap_{x \in X} \text{LiLo}(x)) + \min_{x \in X} \text{NoLi}(x) + |\cap_{x \in X} \text{ReSu}(x)|,$$

where $\lambda([\alpha, \beta]) = \beta - \alpha$ and where $|Y|$ is the number of elements in Y , is a multiway dissimilarity measure on E_1 .

Dissimilarity measures play an important role in cluster analysis where they are often used for constructing clusters having a weak within-cluster dissimilarity degree and/or a strong between-cluster dissimilarity degree. Weak clusters introduced in [6] in the framework of pairwise similarity measures are among these clusters. They are said to be weak in contrast to so-called strong clusters defined as follows. A nonempty subset X of E is said to be a *strong* cluster associated with a pairwise dissimilarity measure d_2 (or *d_2 -strong* cluster), if its *d_2 -strong isolation degree*

$$\min_{\substack{x, y \in X \\ z \notin X}} \{d_2(x, z) - d_2(x, y)\}$$

is positive. Figure 1 illustrates the configuration satisfied by a strong cluster associated with a pairwise dissimilarity measure, say d : for all x, y within the cluster and all z outside it, each of the dissimilarities $d(x, z)$ and $d(y, z)$ is greater than the dissimilarity $d(x, y)$.

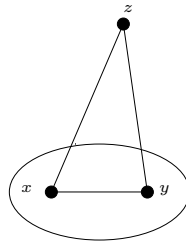


Fig. 1. Strong cluster associated with a pairwise dissimilarity measure

A nonempty subset X of E is said to be a *weak* cluster associated with a pairwise dissimilarity measure d_2 (or *d_2 -weak* cluster), if its *d_2 -weak isolation degree*

$$\min_{\substack{x, y \in X \\ z \notin X}} \{\max\{d_2(x, z), d_2(y, z)\} - d_2(x, y)\}$$

is positive. Figure 2 presents the configuration satisfied by a weak cluster associated with a pairwise dissimilarity measure, say d : for all x, y within the cluster and all z outside it, at least one of the dissimilarities $d(x, z)$ and $d(y, z)$ is greater than the dissimilarity $d(x, y)$.

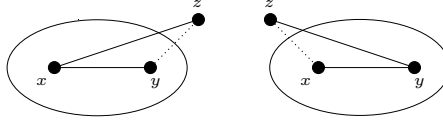


Fig. 2. Weak cluster associated with a pairwise dissimilarity measure

It should be noticed that any d_2 -strong cluster is a d_2 -weak one. Moreover, the notion of weak cluster has been naturally extended to multiway dissimilarity measures [7,13]. A nonempty subset X of E is said to be a *weak cluster* associated with a k -way dissimilarity measure d_k (or d_k -weak cluster) if its d_k -weak isolation degree

$$\min_{\substack{Y \in X_{\leq k}^* \\ z \notin X}} \left\{ \max_{Z \in Y_{\leq k-1}^*} d_k(Z + z) - d_k(Y) \right\}$$

is positive.

4 Relation between Concept Extensions and Weak Clusters

Let a *valuation* on a poset (P, \leq) be any map $h : P \rightarrow \mathbb{R}_+$ such that $h(x) \leq h(y)$ when $x \leq y$ [8]. A *strict index* will be a valuation h such that $x < y$ implies $h(x) < h(y)$. A multiway dissimilarity measure d on E will be said to be *meet-compatible* if there exists a valuation h on \underline{D} with which it is meet-compatible, i.e., such that $d(X) \leq d(Y) \iff h(\inf \delta(X)) \geq h(\inf \delta(Y))$, for $X, Y \subseteq E$. If h is a strict valuation, d will be said to be *strictly meet-compatible*. Meet-compatibility is a kind of natural agreement expressing the following fact, relatively to $(E, \underline{D}, \delta)$: the lower the meet of descriptions of entities in X , the larger the dissimilarity degree of X is. To fix the ideas, assume that a part of entity description space is that depicted in Figure 3. Then any meet-compatible 2-way dissimilarity measure d must satisfy the following inequalities: $d(x, u) \leq d(y, u) = d(x, y) \leq d(u, v) = d(x, v)$ and $d(y, v) \leq d(u, v)$.

Remark 1. The reader may observe that when \underline{D} is a join-semilattice, a dual compatibility condition, say join-compatibility, can be defined by reversing the right-hand side inequality in the above equivalence and replacing meets by joins.

Example 2. The multiway dissimilarity measure given in Example 1 is strictly meet-compatible. Indeed, λ , $x \mapsto x$ and $Y \mapsto |Y|$ are strict valuations on $(CI([0, 24]), \subseteq)$, $([30; 40], \leq)$ and $(\mathcal{P}(S), \subseteq)$, respectively. Then h_1 defined by

$$h_1(u, v, w) = \lambda(u) + v + |w|$$

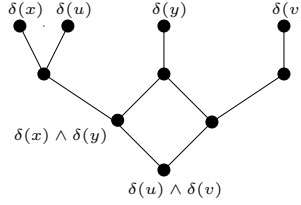


Fig. 3. A part of entity description space

is a strict valuation on \underline{D}_1 , and the fact that dis is meet-compatible with h_1 follows from the fact that $\text{dis}(X)$ is decreasing w.r.t. $h_1(\inf \delta_1(X))$.

Recall that the breadth of a meet-semilattice (P, \leq) is the least positive integer k such that the meet of any $(k + 1)$ elements of P is always the meet of k elements among these $k + 1$ [10]. Having noticed this, we agree to say that a subset Q of a meet-semilattice is of breadth k if k is the least positive integer such that for any $(k + 1)$ -element subset X of Q there is $x \in X$ with $\inf(X - x) \leq x$. Then we have the following result.

Theorem 1. *Let X be a nonempty subset of E and, for integers $p \geq 2$, let d_p be a strictly meet-compatible p -way dissimilarity measure on E .*

- (a) *If $\delta(E)$ is of breadth 1, then the following conditions are equivalent.*
 - (a1) *X is a $(E, \underline{D}, \delta)$ concept extension, i.e. $\phi_\delta(X) = X$.*
 - (a2) *X is a d_2 -strong cluster.*
- (b) *If $\delta(E)$ is of breadth $k \geq 2$, then the following conditions are equivalent.*
 - (b1) *X is a $(E, \underline{D}, \delta)$ concept extension.*
 - (b2) *X is a d_k -weak cluster.*

Proof. (a) Let $\phi_\delta(X) = X$ and let $x, y \in X, z \notin X$. Then, as $\delta(E)$ is of breadth 1, $\delta(z) < \inf \delta(x) \wedge \delta(y)$, so that $d_2(x, z) > d_2(x, y)$ since d_2 is strictly meet-compatible. This proves that X is a d_2 -strong cluster. Conversely, let $x \in X$ such that $\inf \delta(X) = \delta(x)$ and let $z \notin X$. Then $d_2(x, z) > d_2(x)$ so that we don't have $\delta(x) \leq \delta(z)$, proving that $\phi_\delta(X) = X$. Assertion (b) proves with similar arguments. \square

It may be observed that when $\delta(E)$ is of breadth 1, weak clusters associated with a strictly meet-compatible pairwise dissimilarity measure coincide with strong clusters associated with this dissimilarity measure. Then, as E is assumed to be finite, the following result can be derived from Theorem 1.

Corollary 1. *There is an integer $k \geq 2$ such that a nonempty subset of E is a $(E, \underline{D}, \delta)$ concept extension if and only if it is a weak cluster associated with a strictly meet-compatible k -way dissimilarity measure.*

It should be noticed that, by duality principle, when \underline{D} is a join-semilattice, similar results hold between nonempty $(E, \underline{D}, \delta)$ dual concept extensions and weak clusters associated with strictly join-compatible multiway dissimilarity measures.

5 Conclusion

We have outlined an exact relation between (dual) concepts assigned to some contexts and clusters associated with some pairwise or multiway dissimilarity measures. This result, which particularly holds for concepts of standard formal contexts, appears to us as a real bridge between cluster analysis and FCA. Indeed, on the one hand, contexts and their assigned concepts are the basic notions of FCA and, on the other hand, the notion of a cluster is central in cluster analysis. Besides this connection, the result suggests a quantitative concept selection criterion as well as a new way for constructing clusters having a conceptual description. Moreover, the (multiway) dissimilarities measures in question can be easily derived from a context $(E, \underline{D}, \delta)$, and their properties lead to the following formula which can be of help for efficient computation of concepts of the given context: the least concept extension containing a subset X of E is

$$\phi_{\delta}(X) = \{x \in E : h(\inf \delta(X \cup \{x\})) = h(\inf \delta(X))\},$$

where h is any strict valuation on \underline{D} . A similar formula can be found in [14] where the authors use a so-called weight compatible with a closure operator.

To get an idea of the potential usefulness of our result, let us just consider the following application where (semi-automatic) identification of objects in legacy code is addressed, using both cluster analysis and FCA methods [15]. For this application, constructing weak clusters associated with some strictly meet-compatible multiway dissimilarity measure will obviously preserve benefits of separate use of both cluster analysis and FCA methods, while avoiding such problems as those related by the authors: (1) non overlapping of clusters produced by an agglomerative hierarchical method, (2) dependence of the output on the choice of clusters to be merged when there are more than one closest cluster, (3) dendrograms difficult to interpret. Moreover, one can restrict himself in computing only clusters generated by selected subsets (for instance, with at most a given number of elements) and/or with isolation degree within some fixed interval. This can result in a proper subset of the concept lattice, which may be of some interest from an informative viewpoint.

References

1. Brito, P.: Order structure of symbolic assertion objects. *IEEE Transactions on Knowledge and Data Engineering* **6** (1994) 830–835
2. Polaillon, G.: Interpretation and reduction of Galois lattices of complex data. In Rizzi, A., Vichi, M., Bock, H.H., eds.: *Advances in Data Science and Classification*, Springer-Verlag (1998) 433–440
3. M. Liquière and J. Sallantin: Structural machine learning with Galois lattice and graphs. In: *Proc. 5th Int. Conf. Machine Learning ICML'98*, Morgan Kaufman (1998) 305–313
4. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. *Lecture Notes in Computer Science* **2120** (2001) 129–142

5. Diatta, J., Ralambondrainy, H.: The conceptual weak hierarchy associated with a dissimilarity measure. *Mathematical Social Sciences* **44** (2002) 301–319
6. Bandelt, H.J., Dress, A.W.M.: Weak hierarchies associated with similarity measures: an additive clustering technique. *Bull. Math. Biology* **51** (1989) 113–166
7. Bandelt, H.J., Dress, A.W.M.: An order theoretic framework for overlapping clustering. *Discrete Mathematics* **136** (1994) 21–37
8. Barbut, M., Monjardet, B.: *Ordre et classification*. Hachette, Paris (1970)
9. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In Rival, I., ed.: *Ordered sets*. Ridel, Dordrecht-Boston (1982) 445–470
10. Birkhoff, G.: *Lattice theory*. 3rd edition, Coll. Publ., XXV. American Mathematical Society, Providence, RI (1967)
11. S. Joly and G. Le Calvé: Three-way distances. *J. Classification* **12** (1995) 191–205
12. Bennani, M., Heiser, W.J.: Triadic distance models: axiomatization and least squares representation. *J. Math. Psychology* **41** (1997) 189–206
13. Diatta, J.: Dissimilarités multivoies et généralisations d’hypergraphes sans triangles. *Math. Inf. Sci. hum.* **138** (1997) 57–73
14. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with TITANIC. *Data & Knowledge Engineering* **42** (2002) 189–222
15. van Deursen, A., Kuipers, T.: Identifying objects using cluster and concept analysis. Technical Report SEN-R981, CWI, Amsterdam, The Netherlands (1984)

Unlocking the Semantics of Roget's Thesaurus Using Formal Concept Analysis

L. John Old

School of Computing, Napier University
j.old@napier.ac.uk

Abstract. Roget's Thesaurus is a semantic dictionary that is organized by concepts rather than words. It has an elaborate implicit structure that has not, in the 150 years since its inception, been made explicit. Formal Concept Analysis (FCA) is a tool that can be used by researchers for the organization, analysis and visualization of complex hidden structures. In this paper we illustrate two ways in which FCA is being used to explicate the implicit structures in Roget's Thesaurus: implications and Type-10 chain components.

1 Introduction

Like The Bible and Shakespeare, Roget's Thesaurus, for English speakers, is a cultural artefact. School children are taught how to use it at school and it is found on educated English speaker's bookshelves. In real life it is mainly used for crossword puzzles, for finding synonyms to avoid repetition in written work, or to find out what a word means by viewing the company it keeps in the Thesaurus. Whatever its use, it is acknowledged to be a rich source of "meaning."

Roget's Thesaurus has been studied or used for the automatic classification of text, automatic indexing, natural language processing, word sense disambiguation, semantic classification, computer-based reasoning, content analysis, discourse analysis, automatic translation, and a range of other applications. This research has involved mainly the American edition, or Roget's International Thesaurus (RIT), and usually the 3rd Edition (Berrey, 1962). Roget's Thesaurus was also used as the basis for WordNet (Miller, G., Beckwith, Fellbaum, Gross, Miller, K., & Tengi, 1993), the electronic model of the mental lexicon.

For researchers the dream of actually capturing and utilizing the semantics, or meaning, of the word associations in Roget's has been elusive. In part this has been because of a lack of a visualization method that would allow the analysis and insights that would unlock the "inner structure" (Sedelow, W. A. Jr., 1990). Formal Concept Analysis (Wille, 1982) has the ability to automatically classify and arrange information while retaining the complete data, to produce graphics of lattices (Hasse diagrams), and to make relational structures explicit. This gives researchers, we believe, the tool to unlock the inner structure of words and senses in Roget's Thesaurus.

2 The Structure of Roget's Thesaurus

Roget's Thesaurus's organizational structure is a classification tree, or conceptual hierarchy. At the lowest level are what is commonly known as synonyms. The explicit

structure of the book consists of three main parts. Following the front matter is the top level of the hierarchy represented by the tabular Synopsis of Categories. This is followed by the body, or Sense Index of the book, which continues the hierarchy down to the lowest level. The Sense Index lists the 1,000 or so Categories (also called headwords, or lemmas, by some researchers) representing the notions found at the most detailed level of the "Synopsis." Categories generally occur in pairs as opposed notions, or antonyms. Each Category contains the entries¹ – instances of words ordered by part-of-speech and grouped by sense, or synset² (Miller et al., 1993). Synsets exist in groups of senses within a Category, so an index such as 227:1:1 is used, where 227 is the Category; the second number indexes the sense group or *Paragraph* within the Category; and the third number represents the sequence number of the synset, or sense, within that group. At the back of the book is the Word Index, listing the words in alphabetic order, along with their senses ordered by part- of-speech. The senses are represented in the Word Index as references to locations in the Sense Index. On any particular page of the Sense Index the relations of synonymy and antonymy can be seen. By tracing a word out from its synonyms, cross-references, or antonyms to its distant neighbours, all facets of the meaning or semantics of a word can be derived.

The explicit structure of Roget's thesaurus is evident to any reader. The implicit, hidden, or inner structure is not. The relations between instances of a word located in separate parts of the Sense Index, or senses located at separate places in the Word Index, can be made explicit only by automated means. Formal Concept Analysis (FCA) is a natural fit for both analysis and visualization of Roget's Thesaurus. Section 3 describes a formalization of Roget's Thesaurus. Sections 4 and 5 illustrate examples of the application of FCA to words and senses from Roget's International Thesaurus, 3rd Edition (1963).

3 Formalizing Roget's Thesaurus with FCA

Several researchers have used so-called neighbourhood lattices to visualize parts of Roget's thesaurus. The original formalization was suggested by Wille in an unpublished manuscript. Priss (1996) defines neighbourhood lattices as follows:

Instead of using the prime operator ($'$), the plus operator ($+$) retrieves for a set of objects all attributes that belong to at least one of the objects. Formally, for a set G_1 of objects in a context (G, M, I) , $\iota^+(G_1) := \{m \in M \mid \exists_{g \in G_1} : gIm\}$. Similarly $\varepsilon^+(M_1) := \{g \in G \mid \exists_{m \in M_1} : gIm\}$ for a set M_1 of attributes. If two plus mappings are applied to a set G_1 it results in a set $\varepsilon^+\iota^+(G_1)$ (with $\varepsilon^+\iota^+(G_1) \supseteq G_1$) which is called the *neighbourhood* of G_1 under I . A neighbourhood of attributes is defined analogously. A neighbourhood context is a context whose sets of objects and attributes are neighbourhoods, such as $(\varepsilon^+\iota^+(G_1), \iota^+\varepsilon^+\iota^+(G_1), I)$. The resulting lattice is called a neighbourhood lattice.

¹ An entry is a particular sense of a particular word. In this way Synset 227:1:1-Covering contains one of the twenty-two senses of the word *over*. Synset 36:13:1-Superiority contains another instance of *over*. The two occurrences of *over* are two separate entries in Roget's Thesaurus; but only one *word*.

² When referring to the sets of synonyms, the term *synset* will be used. When the *meaning* represented by the words in the Synset is referred to, the term *sense* will be used.

4 Semantic Containment between Words

A word is called semantically contained in another word if the set of Synsets of the first word is a subset of the set of Synsets of the second word. In this case, the semantically contained word is more specific and *implies* the second. This Section shows an example of semantic containments among words from RIT. A sample of words from RIT, between which semantic containment exists, is given in Table 1. The words on the right are semantically contained in the words on the left. A containment relation is always a true subset, not equal, in order to exclude perfect synonyms (words that share every sense, and only those senses).

Table 1. Semantic containment among some pairs of words

SuperSet	SubSet	SuperSet	SubSet	SuperSet	SubSet
3-D	stereoscopic	allowance	stipend	brief	short and sweet
abandoned	deserted	bloody	gory	calm	tranquil
about	circa	blunt	take the edge off	caustic	escharotic
allow	deem	blush	turn red	cave	grotto

The examples in Table 1 are pairs (just twelve of about 10,000). A graph of all semantic containment relations shows an elaborate semantic topology. The semantic containments between words are, likewise, much more elaborate. For example *twaddle*, which has four senses occurring in two Categories, 545: Meaninglessness and 594: Talkativeness, shares these senses with *babble* and *jabber*. *Babble* and *jabber* have other senses in Categories 472: Insanity, Mania, and 578 Language, respectively (amongst others). Furthermore, there are words, such as *gibble-gabble* and *twattle* that are perfect synonyms of *twaddle* because they have exactly the same senses as *twaddle*. There are a number of words that share an intermediate number of senses between *twaddle*, and *babble* and *jabber*, such as *prate* and *gibber*. And there are words that share senses with various combinations of the given words, although this example will omit those from the discussion in order to reduce the complexity of the relationships, and the discussion. All of this is visible in the lattice in Figure 1, which was automatically derived from RIT.

The lattice in Figure 1 is not a complete neighbourhood lattice of any of the words. There are a further 51 words involved in the full lattice of Figure 1 if the semantic neighbourhood of *twaddle*, alone, is taken into consideration. Those words, such as *blether*, *chatter*, *palaver*, *gush*, *spout*, and *gab* form more-complex relationships where, for example, pairs of words can form semantic containments. If all of the words from the eighteen senses in Figure 1 were included, 93 words in all would be pulled in as neighbours. Only an automated method, as is facilitated by FCA, can deal with this type of semantic complexity.

5 Semantic Components in Roget’s Thesaurus

This Section examines the automatically derived *and implicit* Type-10 chains and Components of the mathematical model of abstract thesauri, of which Roget’s Thesaurus

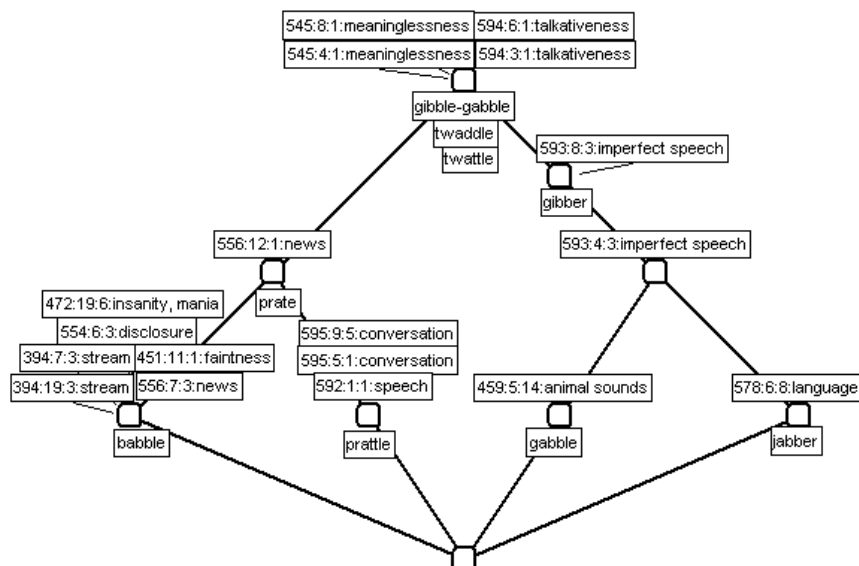


Fig. 1. A lattice showing semantic containment in RIT

is one instantiation, developed by Bryan (1991). The elements in this model are word-strings – which may be single words, compound words, or phrases – and senses (sense definitions, or Synsets), and a relation between them. Bryan defined a series of chains linking the entries by word associations, sense associations, or both. The most restrictive, the Type-10 chain, is a double chain, which requires at least two words to share a sense or two senses to share a word (dubbed a *Quartet*), in order to participate in the chain. This ensures that links are not arbitrary, as happens when two senses are linked by homographs (identical spellings but with disjoint meanings) such as *lead* (the metal) and *lead* (to command).

Talbur and Mooney (1990) derived all possible Type-10 chain Components from the 200,000 entries in RIT and found that the majority of semantically strong connectivity formed one large component network of sense-sense and word-word Quartets. This was partitioned from 5,960 lesser component networks; and was dubbed TMC-69 (Talbur-Mooney Component number 69). TMC-69 included 22,242 inter-linked entries. Jacuzzi (1991) reduced this network by applying a further constraint – that a Quartet could not participate in a component if it shared only one entry with that component. TMC-69 was consequently reduced to 2,507 smaller Jacuzzi components, the largest, of 1,490 entries composed from 324 senses and 312 words, was dubbed VJC-184 (V.Jacuzzi Component number 184). The second largest Component was VJC-1501, with 705 entries. While TMC-69 was a massive inter-connected network of word and sense associations, the resulting derived VJC-184 is a small, but extremely densely bound network – a core of the core connectivities of the semantics of Roget's Thesaurus, and of the English language.

5.1 The Semantics of VJC-184

The most prominent semantic features of VJC-184 emerge not from the numbers but from the Synsets (Synset Category labels) and words. The top most frequent Categories are 855: Excitement, 394: Stream, and 323: Agitation. On first sight, "stream" appears to be semantically incongruent with *excitement* and *agitation* because it lies between Categories 393: Rain and 395: Channel in the RIT Synopsis of Categories and is certainly a classifier of "water" words. But the semantic relationship becomes clearer on closer inspection. Some of the 29 words occurring in VJC-184 that are derived from senses in Category 394: Stream, in descending order of frequency, are *flood*, *gush*, *run*, *surge*, *flow*, *deluge*, *rush*, *race*, and *course*. These words, used in their metaphoric, non-liquid senses are in fact congruent with *excitement* and *agitation*.

Table 2. Top 20 words in RIT by polysemy

Word	Polysemy	in VJC-184	in VJC-1501	Word	Polysemy	in VJC-184	in VJC-1501
cut	64		x	sound	37		
run	54	x		break	36		x
set	51	x		check	36		
turn	45	x		discharge	36		x
head	43	x		drop	35		
pass	41			cast	34		
charge	41			go	34	x	
close	39			lead	34	x	
line	38	x		light	34		
beat	37	x	x	form	34	x	

The top two senses – those contributing the most words to entries (and therefore the Quartets, and connectivity) in VJC-184 – are: 323:1:1 {*fume*, *bluster*, *bustle*, *churn*, *commotion*... } (of 30 synonyms), and 62:4:1 {*row*[*argue*], *bluster*, *bother*, *brawl*... } (of 28). The overall most frequent words (out of the total of 312 words) begin, in order of frequency: *turn* (30 entries), *course* (20), *run* (18), *clash* (15), *bother* (15)... These words correspond closely to the top twenty words, by polysemy (number of senses the word has), in RIT.

The top (most polysemous, and therefore most frequent) twenty words in RIT (of 113,000 total) are listed in Table 2. The table indicates whether the words are found in VJC-184 or in VJC-1501 (the second largest sub-component of TMC-69). Some of the words are found in other Components. The word "beat" occurs in both VJC-184 and VJC-1501³.

An intuition about the semantics may be gained from listing these most frequent elements of VJC-184, but a different method is necessary to gain insights into the relationships among the elements. Figure 2 is a lattice of VJC-184. To reduce the complexity only words and senses that occur in at least ten entries of Component VJC-184 are included.

³ VJC-1501 is characterized by words such as *cut*, *crack*, *hit*, *bust*, *gash*, *split*, *break*.

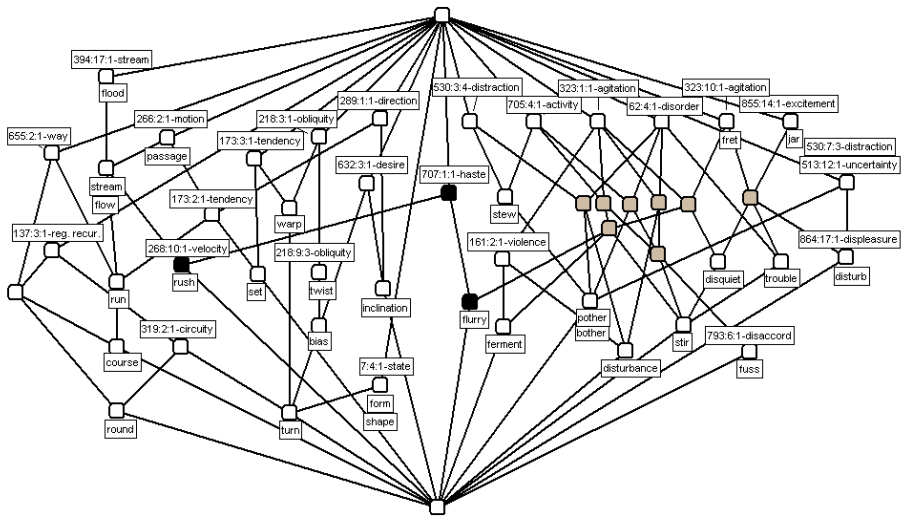


Fig. 2. Lattice of VJC-184 restricted to senses and words with ten or more instances (contributing to at least ten entries in RIT). Also visible are the “motion” cluster (left) and the “commotion” cluster (right)

There is a clear left-right division within VJ-184, connected in the middle by Synset 707:1:1 Haste (the label above the top black-filled Formal Concept), through the sharing of *rush* (left black Formal Concept) and *flurry* (right black Formal Concept). The left collection has semantics characterized by *turn, run, course*/Stream, Motion, Direction. The right hand collection has semantics characterized by *fuss, bother, trouble*/Agitation, Excitement, Disorder. For brevity they will be referred to respectively as the “motion” and “commotion” groups, or clusters. The motion and commotion groups of VJC-184 would have been extremely difficult to detect without the aid of a lattice diagram arrangement of the words and senses. Lindig and Snelting’s (1997) work on horizontal decomposition of lattices offers an algorithm solution to identifying such divisions. Table 3 shows how Synset 707:1:1 ties together the motion and commotion groups.

5.2 Unlabeled Concepts

The commotion group (right side of the lattice) displays a feature hidden by any other form of representation. When at least two objects share two attributes in a Formal Concept lattice, but in addition each of the four elements is differentiated by further objects and attributes that are not shared among the other three elements, an “unlabeled Concept” emerges. These are the Concepts coloured in grey, in Figure 2. While unlabeled Concepts are always rare in lattice representations of semantic data, an entire cluster of unlabeled Concepts is has not been observed elsewhere. The cluster of unlabeled concepts in the commotion group suggests a large number of words with overlapping hues and tones of meaning, discriminated at the edges but not in the center.

Four of the words in the commotion group that contribute to the emergent unlabeled Concepts are *flurry*, *ferment*, *fuss*, and *stir*. These all classify under 705:4:1 Activity,

Table 3. Subcontext showing the connections (via Synset 707:1:1 Haste) between the motion (left) and commotion (right) groups of VJC-184

	Motion	Velocity	Stream	Haste	Agitation	Agitation	Activity
	266:2:1	268:10:1	394:17:1	707:1:1	323:1:1	323:10:1	705:4:1
bustle				x	x		x
flutter				x			x
flurry				x	x	x	x
rush	x	x	x	x			
scamper		x		x			
scramble		x		x			
scurry		x		x			
scuttle		x		x			
dash		x		x			

and 3:1:1 Agitation Those words are also classified under other senses, but in different combinations (subsets); or independently of each other, alongside other words. *Flurry* is also found in 707:1:1 Haste alongside *rush*; and *ferment* is found in 161:2:1 Violence, alongside *disturbance*, for example.

Synset 323:1:1 Agitation holds sway over the majority of unlabeled Concepts. Synset 3:1:1 contributes the most entries to VJC-184 (all nouns). 323:1:1 is, however, not the key to the nest of unlabeled Concepts which is further linked to many, many other words and senses omitted from Figure 2, and which also hold this mesh together. If Synset 323:1:1 (or any individual Synset) is removed, other senses such as 161:2:1 Violence, 705:4:1 Activity, 62:4:1 Disorder, and 323:10:1 Agitation (the verb contribution from Category 323) would continue to hold the structure in place. Like a single strand plucked from a spider's web, the web distorts but mostly holds in place – similarly if words are removed. Further discussion on unlabeled concepts, related to multiple inheritance in class hierarchies, may be found in Godin & Mili (1993).

Some of the dense connections seen in the VJC-184 (and other Components) are comprised of apparently etymologically unrelated words that in fact share common Indo-European roots. Examples from VJC-184 are: *flood*, *fluster*, *flutter*, *flight*, and *flow*, which all derive from the root, PLEU-, meaning “flow.” *Warp*, *pervert*, *wring*, and *wrench* all derive from the root, WER-3, meaning “turn, bend”-and there are others. Such ancient etymological threads may explain why some Synsets are so large, and why they inter-connect so readily. Etymology alone can't explain the cluster of unlabeled Concepts, however, as no single Indo-European root pervades that group. The underlying concept perhaps can be explained by proposing that it is an ancient concept at the root of human conceptual organization-not the central source, although it can be traced out to connect to more than 70,000 entries, but one of several primitive concepts possibly more felt than intellectualised, and a facet of consciousness connected to many other areas of thought.

6 Conclusion

We have shown that Formal Concept Analysis is a tool that can make explicit the implicit relationships in complex data. Roget's Thesaurus as an instantiation of what: “might be

accurately regarded as the skeleton for English-speaking society's collective associative memory" (S. Y. Sedelow, 1991, p.108). Insights into this semantic store can have implications for psychology and cognitive science, linguistics, and even anthropology. This will not be possible without the ability to automatically derive and visualize the implications, semantic neighbourhoods and implicit structures among the semantic elements in Roget's Thesaurus. Formal Concept Analysis is a flexible tool capable of facilitating this process.

References

1. Berrey, L. (Ed.). (1962). *Roget's international thesaurus* (3rd ed.). New York: Crowell.
2. Bryan, R. M. (1973). Abstract thesauri and graph theory applications to thesaurus research. In S. Y. Sedelow (Ed.), *Automated language analysis, report on research 1972-73* (pp. 45-89). Lawrence, KS: University of Kansas.
3. Godin, R. & Mili, H. (1993). Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices. In A. Paepcke (Ed.), *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'93)*, (pp. 394-410). Washington, DC: ACM Press.
4. Jacuzzi, V. (1991, May). Modeling semantic association using the hierarchical structure of Roget's international thesaurus. Paper presented at the Dictionary Society of North America Conference, Columbus, Missouri.
5. Lindig, C., & Snelting, G. (1997). Assessing Modular Structure of Legacy Code Based on Mathematical Concept Analysis. *Proceedings of the 19th International Conference on Software Engineering (ICSE 97)*, Boston, USA, pp. 349-359.
6. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K., & Teng, R. (1993). Five papers on WordNet. Technical Report. Princeton, N.J: Princeton University.
7. Priss, U. (1996). *Relational Concept Analysis: Semantic structures in dictionaries and lexical databases*. (Doctoral Dissertation, Technical University of Darmstadt, 1998). Aachen, Germany: Shaker Verlag.
8. Sedelow, S.Y. (1991). Exploring the terra incognita of whole-language thesauri. In R. Gamble & W. Ball (Eds.), *Proceedings of the Third Midwest AI and Cognitive Science Conference* (pp. 108-111). Carbondale, IL: Southern Illinois University.
9. Sedelow, W. A., Jr. (1990). Computer-based planning technology: an overview of inner structure analysis. In L. J. Old (Ed.), *Getting at disciplinary interdependence*, (pp. 7- 23). Little Rock, AR: Arkansas University Press.
10. Talburt, J. R., & Mooney, D. M. (1990). An evaluation of Type-10 homograph discrimination at the semi-colon level in Roget's international thesaurus. *Proceedings of the 1990 ACM SIGSMALL/PC Symposium*, 156-159.
11. Wille, R., (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, (Ed.), *Ordered sets* (pp. 445-470). Dordrecht: Reidel.

FCA in Knowledge Technologies: Experiences and Opportunities

Yannis Kalfoglou¹, Srinandan Dasmahapatra¹, and Yun-Heh Chen-Burger²

¹ Advanced Knowledge Technologies (AKT), School of Electronics and Computer Science
University of Southampton, Southampton SO17 1BJ, UK

{y.kalfoglou, sd}@ecs.soton.ac.uk

² Advanced Knowledge Technologies (AKT), Artificial Intelligence Applications Institute
School of Informatics, University of Edinburgh, Edinburgh EH8 9LE, UK
jessica@inf.ed.ac.uk

Abstract. Managing knowledge is a difficult and slippery enterprise. A wide variety of technologies have to be invoked in providing support for knowledge requirements, ranging from the acquisition, modelling, maintenance, retrieval, reuse and publishing of knowledge. Any toolset capable of providing support for these would be valuable as its effects would percolate down to all the application domains structured around the domain representation. Given the generic structure of the lattice building algorithms in Formal Concept Analysis, we undertook a set of experiments to examine its potential utility in knowledge technologies. We elaborate on our experiences and speculate on the opportunities lying ahead for a larger uptake of Formal Concept Analysis approaches.

1 Introduction

A distinguishing feature of much of the knowledge technologies today is the attention paid to representations of appropriate domain knowledge as scaffolding around which automated and mixed-initiative processing systems are constructed to provide the required functionality. In particular, with the promise of a Semantic Web a great deal of effort is being directed at providing knowledge-level characterisations of both domains and functionality of processes to achieve inter-operability in an environment as open and distributed as the Web.

These characterisations are significant for representing and modelling domain knowledge in sound and machine-processable manners. Advanced Knowledge Technologies (AKT)¹ is a large interdisciplinary research collaboration between five UK universities working on developing technologies to address these issues. There are a number of technologies used for knowledge-level domain characterisations, namely ontologies, and tools to support their engineering. However, there is little support for the modeller to help in identifying appropriate conceptual structures to capture domain semantics. Formal Concept Analysis (FCA)[8] provides a fertile ground for exploitation with its generic structure of lattice building algorithms to visualize the consequences of partial order that the underlying mathematical theory builds on.

In this paper, we elaborate on our experiences with using FCA in conjunction with knowledge technologies in the context of the AKT project. We also identify joint points

¹ Accessible online from www.aktors.org

where prominent knowledge technologies, like Description Logics for the Semantic Web, could benefit from FCA. We do this in a speculative fashion in the next section, coupled with our example cases to show where possible collaboration could be achieved. We briefly report on similar work in section 3 and conclude the paper in section 4.

2 Experiences and Opportunities for FCA

FCA has been applied at various stages of a system's life cycle: for example, in the early stages when analysing a domain for the purpose of building and using a knowledge-rich representation of that domain - like the work of Bain in [3] where FCA was used to assist building an ontology from scratch - or applied at later stages in order to enhance an existing system for the purpose of providing a specific service - like the *CEM* email management system described in [6].

It appears though that is being used selectively and opportunistically. The reason for this scattered area of FCA applications could be the fundamental ingredients of FCA and its underlying philosophy: FCA emerged in the 80s as a practical application of lattice theory. The core modelling ingredients underpinning FCA are *objects* and *attributes*² which stem from predicative interpretations of set theory. Thus, for a given object, one performs a "closure" operation to form a set of objects which is the intersection of the extension of the attributes that the object is characterised by. These are defined as the concepts in any particular formal context, with the order ideal (or down set) $\downarrow m$ of any attribute m .

In the AKT project, we experimented with identifying these concepts in a number of scenarios from the scientific knowledge management realm where we confronted with loosely defined objects and attributes. Our aim was to use FCA to help us identify the prominent concepts in the domain at question, and most importantly, provide us with a structured representation which we could use to perform certain knowledge management tasks, such as:

Analysing Programme Committee Memberships: One could assume that programme committee membership for a conference or similar event requires that those on the programme committee (PC) are the current and prominent figures in the field at question. Using this as a working hypothesis, and the year in which they served at a specific PC as temporal marker of recognized prominence, we then applied FCA techniques like concept lattice exploration to visualize the distribution of PC members over a number of years. This could, arguably, give us an idea of how the specific event evolved over a period of time by virtue of the changes (or otherwise) in their PCs.

In our experiments (briefly described online in [11]), the objects were PC members and attributes were EKAW conferences in which these members served. A visual inspection of sort of lattice can reveal trends in how the event has evolved over the years. For example, we can identify people who were in PCs of early EKAWs but are not appearing in more recent EKAWs, whereas others have a sustainable presence in the

² Priss points out in [13] these can be *elements*, *individuals*, *tokens*, *instances*, *specimens* and *features*, *characteristics*, *characters*, *defining elements*, respectively.

PCs throughout the whole period of 1994 to 2002. If we correlate this information with information regarding the research interests of the PCs, we could end up with a strong indication of the evolution of research themes for the EKAW conferences. In what we present, we regard the extraction of research interests of these researchers as peripheral to our discussion but we point the interested reader to the work done on identifying communities of practice in [2].

Analysing the Evolution of Research Themes: This analysis can be supported by another lattice which depicts the evolution of research themes in EKAW conferences, based on the designated conference session topics. We depict this lattice in figure 1. From the lattice drawing point of view, we should note that we used a publicly available FCA tool, *ConExp*³ but we deliberately changed the position of the nodes in the line diagrams produced. We did that to enhance its readability and ease its illustration when depicted on paper as we wanted to include all labels from objects and attributes. That compromised the grid projection property of the diagram without, however, affecting the representation of partial order between nodes.

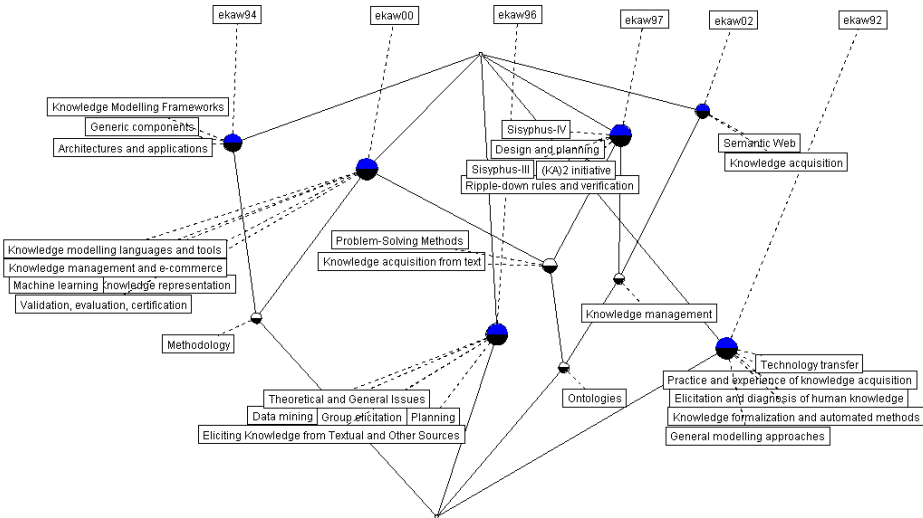


Fig. 1. Concept lattice depicting session topics of the EKAW conferences from 1994 to 2002.

Again, a close inspection shows some trends which are evident in today's research agendas in many organisations: *knowledge modelling frameworks* and *generic components* were popular in the early 90s whereas nowadays the research focus is on *semantic web* and *knowledge management*. The inherited taxonomic reasoning of concept lattices can also reveal interesting relationships between research topics, as for instance the subsumption of *ontologies* from *knowledge management*, *knowledge acquisition* and the *semantic web* topics.

³ Presented in [17] and can be downloaded from <http://sourceforge.net/projects/conexp>

Analysing Research Areas Attributed to Published Papers: We also applied FCA techniques, in particular context reduction algorithms like those described in the FCA textbook (p.27 in [8]) to analyse the formal context of online academic journals. Our aim was to expose relationships between research areas used to classify published papers. The premise of our analysis is the very algorithm that Ganter and Wille describe in [8] for clarifying and reducing formal contexts: “[...] we merge objects with the same intents and attributes with the same extents. Then we delete all objects, the intent of which can be represented as the intersection of other object intents, and correspondingly all attributes, the extent of which is the intersection of other attributes extents.”. This process, if captured in a step-wise fashion, will expose the objects and attributes that are about to be merged with others, hence allowing us to infer that they are related.

For our data sets, we used a small number of articles from the *ACM Digital Library* portal⁴ focusing on the *ACM Intelligence* journal⁵. The formal context consists of 20 objects (articles) and 58 attributes (research areas). The research areas originate from a standard classification system, the *ACM Computing Classification System*⁶. We also used a second data set, the *Data and Knowledge Engineering (DKE)* journal from Elsevier⁷. In this context we had the same articles (objects) as in the ACM context, but this time we classified them against the *DKE*’s own classification system, Elsevier’s classification of *DKE* fields⁸, which used 27 research areas (attributes) for classifying the aforementioned articles.

For both data sets we chose as objects for their context papers that appeared in the journals. For instance, for the *ACM Intelligence* journal we chose papers that appeared over a period of three years, from 1999 to 2001 and were accessible from the *ACM Digital Library* portal. As there were already classified according to the *ACM Computing Classification System*, we used their classification categories as attributes. We then applied typical context reduction techniques in a step-wise fashion. While we were getting a reduced context, we captured the concepts that are deemed to be interrelated by virtue of having their extents (objects that represent articles in the journal) intersected. For instance, the *ACM classification category H.3.5 on Web-based services* is the intersection of *H.5 on Information Interfaces and Presentation* and *H.3 on Information Storage and Retrieval* by virtue of classifying the same articles. This sort of analysis supports identification of related research areas using as supporting evidence the classification of articles against standardized categories as those originating from the *ACM Computing Classification System*, and the inherited taxonomic reasoning which FCA concept lattices provides to infer their relationship.

Although these experiments were based on loosely defined objects and attributes, it is evident that knowledge representation formalisms also deal with concepts in as much as they underpin systems that require expressive domain characterisation. Much of this effort involves a restricted predicative apparatus in order that the expressivity allowed by appropriate formalisms does not lead to well-known problems of decidability.

⁴ Accessible online from <http://portal.acm.org>

⁵ Accessible online from <http://www.acm.org/sigart/int/>

⁶ Accessible online from <http://www.acm.org/class/1998/>

⁷ Accessible online from <http://www.elsevier.com/locate/issn/0169023X/>

⁸ Accessible online from

<http://www.elsevier.com/homepage/sac/datak/dke-classification-2002.pdf>

To illustrate the point, recall that the experiments we reported above were conducted within the umbrella of the AKT project which has been developing and applying knowledge technologies for the Semantic Web. In the Semantic Web enterprise ontologies have been identified as a key enabling construct, and Web Ontology Language (OWL) is currently going through the final stages of adoption as the W3C standard for the language of choice in which to express ontologies on the Web. OWL inherits characteristics of Description Logics (DLs) which are languages of restricted expressivity designed to facilitate tractable taxonomic reasoning. Some of the simpler and computationally less expensive logics are fragments of first-order predicate logic with two variables, small enough to express binary predicates[4]. A binary predicate can also be described graphically as an edge connecting two nodes each representing a variable, and the knowledge modelling experiments reported below use such a diagrammatic representation.

AKT Research Map: We obtained these kind of diagrams for the same domain as the the one described in the experiments reported above during a knowledge modelling exercise, the construction of the *AKT Research Map*[5], which would capture the expertise and areas of research conducted by AKT members. The specific case was to plan, build, check and publish the *AKT Research Map* and its ontology which is an extension of the *AKT Reference* ontology[1]. The motivation of producing such a map is to fill the knowledge gap that will not normally be filled by methods such as automatic information extraction techniques from Web pages. While the Web is very good in advertising the results of research work, such as publications and white papers, it is also poor in publishing the in-depth knowledge about how such research results were produced, what sort of resources were utilized, what contacts, best practices, etc. This information, however, is valuable for other researchers who have similar interests. To address this problem, an *AKT Research Map* has been built to capture such in-depth know-how as well as collaborators, software systems and literature that are contributors to forming the final result.

The *AKT Research Map* is written using a modelling method that is a specialization of Entity Relational Data Modelling method[15]. As a part of building the *AKT Research Map*, members of the AKT project participated in structured interviews to develop their own individual maps during knowledge acquisition (KA) sessions by extending concepts and relations expressed in the OWL-based *AKT Reference* ontology. As an example, figure 2 depicts such an individual map of the AKT member Nigel Shadbolt.

Between the KA sessions and when a draft of the *AKT Research Map* was finished, iterative verification and validation was carried out to make sure that the map is consistent within itself and consistent with the underpinning ontology. This includes checking of inconsistency of the same information that has been described in different parts of the map, and that a same object has not been represented under different names. It also carries out a pair-wise model checking between the map and the ontology to make sure that the map ontology is complete and the research map is consistent with the ontology. This process could have been performed using FCA lattice drawing facilities had our tools been so integrated.

In making the comparison with FCA, every attribute column in the cross table of the formal context can be viewed predicatively in the sense of DLs or equivalently in

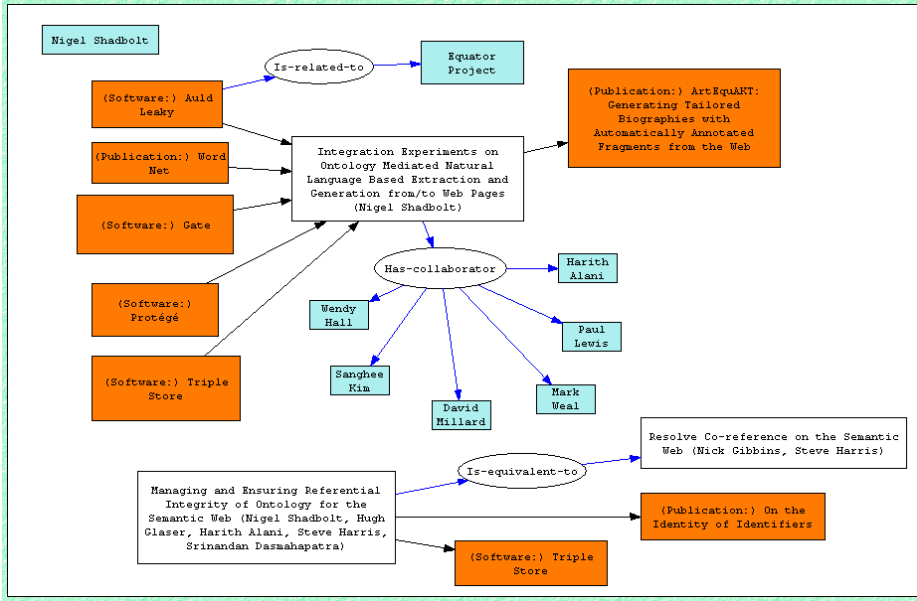


Fig. 2. The *AKT Research Map* of Nigel Shadbolt.

terms of the diagrams in the *AKT Research Map* – a directed edge or a binary relation shows up at the corresponding cell marked by the row and column positions. To make the correspondence work, the formulae in the ontology need to be instantiated by *individuals*. Concept lattices built with FCA makes concept dependencies manifest, thus making it an appealing tool for any modeller. For instance, the fragment of the *AKT Research Map* in figure 2 depicts a link with the label “Has-collaborator” a relation that is not present in the underlying *AKT Reference Ontology*. By choosing the set of people as objects and a limited set of attributes – that of authorship of some individual papers, or being investigators of some individual projects – a concept can emerge (after suitably taking intersections of the appropriate sets) which includes Nigel Shadbolt and all of his collaborators. A modeller can choose to extract a relationship based on this derived concept, which can then be defined into the conceptual structure using the syntax of the logical formalism. This requires lifting the propositions represented into a first-order formalism, i.e., introducing a variable in place of the list of instances upon which the modeller makes her decision.

In DLs, concept labels are assigned a formal semantics by their extension, as sets of objects. These can be primitive or defined by conjunction with other concepts. Binary relationships are used with existential or universal quantifiers to define properties these concepts might possess. So, for example, in a domain Δ , concept C would have as the domain of its interpretation function $\mathcal{I} : \cdot \rightarrow \Delta$ the set $C^{\mathcal{I}} \subset \Delta$. Furthermore, predicative properties are defined by role restriction; hence for binary relation R and concept C , we can create $\forall R.C$ or $\exists R.C$ whose extensions are, respectively,

$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta \mid \forall y(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \quad \text{and} \\ (\exists R.C)^{\mathcal{I}} = \{x \in \Delta \mid \exists y(x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}.$$

Classes are also defined by conjunction and disjunction which facilitates subsumption reasoning of the form $A^{\mathcal{I}} \cap B^{\mathcal{I}} \subseteq A^{\mathcal{I}}$ etc. These enable the modeller to check for consistency of the ontology under development in the cases when concepts are constructed in terms of others.

However, since every attribute is defined *solely* by its extension, concepts generated by join and meet operations on the lattice may not have obvious labels that a domain expert would be comfortable with, a problem associated with any bottom-up approach. Bain[3], notes that “a drawback in FCA is that it does not allow for the introduction of concept names as new intermediate-level terms [...] this is necessary for re-use of conceptual structures, so that they may be referred to by name, e.g. to be used in incremental learning or theory revision.” while Priss records these as being characterised as “informationless” states[13].

In the DL world, the approach to modelling is much more top-down, and ontologies are often created (as terminologies or T-boxes) with no instance information, despite the semantics being defined extensionally (for example, in A-Boxes). This makes direct comparison of FCA lattices with taxonomic trees rendered in DLs approaches difficult. However, inheritance of properties in a taxonomic hierarchy is possible because the *is-a* relationship (set inclusion) is transitive, and one can identify the down set of an attribute in FCA with the range of the corresponding interpretation function in DLs. This points out the restricted nature of different types of relationships that are represented in FCA. It has been noted in the literature that FCA focusses on hypernym hierarchy of concepts but not arbitrary relations among them[14]. The trees drawn to express taxonomic relationships are, of course, the only ones that DLs formalisms typically generate, even though they were designed to formalise semantic nets and ER diagrams that were described in the *AKT Research Map* above. The work of Hahn *et al* [9] who introduced formal structures to express part-whole relationships as taxonomic ones is an interesting avenue to extend the expressivity of relationships captured in FCA. A detailed study relating these formalisms would be valuable to provide methodological support for knowledge representation.

3 Related Work

FCA has been applied in a variety of applications along the whole spectrum of knowledge management with emphasis on the early stages of acquisition, analysis and modelling. For instance, in [12] the authors describe the *KANavigator*, a web-based browsing mechanism for domain-specific document retrieval. Their work is closely related to our experiments on analysing research interests using FCA. However, *KANavigator* is a more generic system aiming not only at navigational aid when browsing a domain but also providing support for incremental development and evolution of the concepts involved. This could be achieved by the open environment in which the *KANavigator* was deployed where users can update the concept lattice with new objects of interest as these emerge. On the other hand, our aim was to simply capture the dependencies between seemingly related research interests as our approach was to perform further analysis and modelling using richer knowledge modelling structures, like ontologies. Identifying related concepts then, was our priority as it was in [10] work where the authors applied FCA as a conceptual clustering tool by using the intentional descriptions

of objects in a domain as indicators of cluster membership. As our KA experience script highlighted in section 2 the results of this sort of analysis need to be combined with other technologies.

There are few examples of combining FCA with other popular knowledge technologies, one of them is the *Troika* approach described in [7]. The authors combined repertory grids, conceptual graphs and FCA in order to overcome the acknowledged KA bottleneck where there exist a plethora of methods, techniques and tools but none of them is adequate enough to carry out its task elegantly without needing the input of another. The authors argued that these three technologies could be combined in a streamlined fashion, where repertory grids are used for acquisition, FCA for analysis and conceptual graphs for representation. The *Troika* approach is a set of algorithmic steps which dictate the order and the way in which these three technologies can be combined.

There have been attempts in the literature to integrate different approaches in order to support knowledge modelling, as indicated above. Further work has been done by Tilley and colleagues [16] who proposed a solution to provide some more interaction with the modeller in order to question and verify the elements of the formal context by editing, removing or adding new ones. In the domain of software engineering they used FCA to model the class hierarchy in a given domain and compared it with a typical software engineering use-cases based modelling approach. Although identifying initial nouns from the textual specification as candidates for objects in the formal context was a laborious and time consuming exercise, the authors praised the value of the approach as it makes clearer and more collaborative the design process.

4 Conclusions

FCA provides a set of tools that allows for formalising a set of informal descriptions of a domain thus providing the basis for ontology building. The model theory of DL formalisms allow for a direct comparison with the sets that form the extents of concepts in FCA. However, the relationship between subsumption hierarchies that are commonplace in ontologies to the partial orders in FCA lattices needs to be explored in detail if FCA techniques are to be extended to assist the knowledge engineer.

Acknowledgements

This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

References

1. AKT. AKT Reference Ontology, <http://www.aktors.org/ontology/> 2003.
2. H. Alani, S. Dasmahapatra, K. O'Hara, and N. Shadbolt. Identifying Communities of Practice through Ontology Network Analysis. *IEEE Intelligent Systems*, 18(2):18–25, Mar. 2003.

3. M. Bain. Inductive Construction of Ontologies from Formal Concept Analysis. In *Proceedings of the 11th International Conference on Conceptual Structures (ICCS'03)*, Springer LNAI 2746, Dresden, Germany, July 2003.
4. A. Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.
5. Y.-H. Chen-Burger. AKT Research Map v.2.0, <http://www.ai.ai.ed.ac.uk/~jessicac/project/akt-map-html/top-level.html>, 2003.
6. R. Cole and G. Stumme. CEM - a Conceptual email Manager. In *Proceedings of the 8th International Conference on Conceptual Structures (ICCS'00)*, Darmstadt, Germany, Aug. 2000.
7. H. Delugach and B. Lampkin. Troika: Using Grids, Lattices and Graphs in Knowledge Acquisition. In *Proceedings of the 8th International Conference on Conceptual Structures (ICCS'00)*, Darmstadt, Germany, Aug. 2000.
8. B. Ganter and R. Wille. *Formal Concept Analysis: mathematical foundations*. Springer, 1999. ISBN: 3-540-62771-5.
9. U. Hahn, S. Schulz, and M. Romacker. Part-Whole Reasoning: A Case Study in Medical Ontology Engineering. *IEEE Intelligent Systems*, 14(5):59–67, Oct. 1999.
10. A. Hotho and G. Stumme. Conceptual Clustering of Text Clusters. In *Proceedings of the Machine Learning Workshop (FGML'02)*, Hannover, Germany, Oct. 2002.
11. Y. Kalfoglou. Applications of FCA in AKT, <http://www.aktors.org/technologies/fca>, 2003.
12. M. Kim and P. Compton. Incremental Development of Domain-Specific Document Retrieval Systems. In *Proceedings of the 1st International Conference on Knowledge Capture (K-Cap'01)*, Victoria, BC, Canada, Oct. 2001.
13. U. Priss. Formalizing Botanical Taxonomies. In *Proceedings of the 11th International Conference on Conceptual Structures (ICCS'03)*, Springer LNAI 2746, Dresden, Germany, July 2003.
14. C. Schmitz, S. Staab, R. Studer, and J. Tane. Accessing Distributed Learning Repositories through a Courseware Watchdog. In *Proceedings of the E-Learn 2002 world conference on e-learning in corporate, government, healthcare & higher education*, Montreal, Canada, Oct. 2002.
15. B. Thalheim. *Entity Relationship Modeling*. Springer, Dec. 1999. ISBN: 3540-6547-04.
16. T. Tilley. A Software Modelling Exercise using FCA. In *Proceedings of the 11th International Conference on Conceptual Structures (ICCS'03)*, Springer LNAI 2746, Dresden, Germany, July 2003.
17. S. Yevtushenko. System of Data Analysis Concept Explorer (in Russian). In *Proceedings of the 7th National Conference on Artificial Intelligence (KII-2000)*, Russia, pages 127–134, 2000.

Applying Formal Concept Analysis to Description Logics

Franz Baader* and Baris Sertkaya**

Theoretical Computer Science, TU Dresden, D-01062 Dresden, Germany
`{baader,sertkaya}@tcs.inf.tu-dresden.de`

Abstract. Given a finite set $\mathcal{C} := \{C_1, \dots, C_n\}$ of description logic concepts, we are interested in computing the subsumption hierarchy of all least common subsumers of subsets of \mathcal{C} as well as the hierarchy of all conjunctions of subsets of \mathcal{C} . These hierarchies can be used to support the bottom-up construction of description logic knowledge bases. The point is to compute the first hierarchy without having to compute the least common subsumer for all subsets of \mathcal{C} , and the second hierarchy without having to check all possible pairs of such conjunctions explicitly for subsumption. We will show that methods from formal concept analysis developed for computing concept lattices can be employed for this purpose.

1 Introduction

The notion of a concept as a collection of objects sharing certain properties is fundamental to both formal concept analysis (FCA) [19] and description logics (DL) [6]. However, the ways concepts are obtained and described differ significantly between these two research areas. In DL, the relevant concepts of the application domain (its terminology) are formalized by *concept descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept constructors provided by the DL language. In a second step, these concept descriptions together with the roles are used to specify properties of *objects* occurring in the domain. In FCA, one starts with a *formal context*, which (in its simplest form) specifies which (atomic) properties are satisfied by the objects. A *formal concept* of such a context is a pair consisting of a set of objects (the extent) and a set of properties (the intent) such that the intent consists of exactly those properties that the objects in the extent have in common, and the extent consists of exactly those objects that share all the properties in the intent.

There are several differences between these approaches. First, in FCA one starts with a complete (but simple) extensional description of a domain, and then derives the formal concepts of this specific domain, which provide a useful structuring of the domain. In DL, the (intensional) definition of a concept is

* Partially supported by National ICT Australia Limited, Canberra Research Lab.

** Supported by the German Research Foundation (DFG, GRK 433/3).

given independently of a specific domain (interpretation), and the description of the objects is only partial. Second, in FCA the properties are atomic, and the intensional description of a formal concept (by its intent) is just a conjunction of such properties. DLs usually provide a rich language for the intensional definition of concepts, which can be seen as an expressive, yet decidable sublanguage of first-order predicate logic.

There have been several attempts toward bridging this gap between FCA and DL. For example, researchers from the FCA community have extended FCA to incorporate more complex properties [39,31,30,33]. The present paper is concerned with bridging the gap from the other direction. We will describe how tools from FCA can be used to support the *bottom-up* construction of DL knowledge bases, as introduced in [8,9]: instead of directly defining a new concept, the knowledge engineer introduces several typical examples as objects, which are then automatically generalized into a concept description by the system. This description is offered to the knowledge engineer as a possible candidate for a definition of the concept. The task of computing such a concept description can be split into two subtasks: computing the most specific concepts of the given objects, and then computing the least common subsumer of these concepts. The *most specific concept* (msc) of an object o (the *least common subsumer* (lcs) of concept descriptions C_1, \dots, C_n) is the most specific concept description C expressible in the given DL language that has o as an instance (that subsumes C_1, \dots, C_n). The problem of computing the lcs and (to a more limited extent) the msc has already been investigated in the literature [12,15,8,9,25,24,23,4,3,2]. Here, we will address two problems that occur in the context of the bottom-up approach.

First, the methods for computing the least common subsumer are restricted to rather inexpressive descriptions logics not allowing for disjunction (and thus not allowing for full negation). In fact, for languages with disjunction, the lcs of a collection of concepts is just their disjunction, and nothing new can be learned from building it. In contrast, for languages without disjunction, the lcs extracts the “commonalities” of the given collection of concepts. Modern DL systems like FaCT [22] and RACER [21] are based on very expressive DLs, and there exist large knowledge bases that use this expressive power and can be processed by these systems [32,36,20]. In order to allow the user to re-use concepts defined in such existing knowledge bases and still support the user during the definition of new concepts with the bottom-up approach sketched above, we propose the following extended bottom-up approach. There is a (background) terminology \mathcal{T} defined in an expressive DL \mathcal{L}_2 . When defining new concepts, the user employs only a sublanguage \mathcal{L}_1 of \mathcal{L}_2 , for which computing the lcs makes sense. However, in addition to primitive concepts and roles, the concept descriptions written in the DL \mathcal{L}_1 may also contain names of concepts defined in \mathcal{T} . When computing subsumption between such newly defined concepts, this is done w.r.t. \mathcal{T} , using a subsumption algorithm for the expressive DL \mathcal{L}_2 . When computing the lcs of such concepts, we basically employ the algorithm for \mathcal{L}_1 , but extend it such that it can take into account the subsumption relationships between conjunctions of

concept defined in \mathcal{T} . This is where FCA comes into play: it provides us with an efficient method for computing these relationships. To be more precise, given a TBox \mathcal{T} , we define a formal context whose properties are the defined concepts of \mathcal{T} , and whose concept lattice is isomorphic to the subsumption hierarchy we are interested in. Then, we employ the so-called “attribute exploration” algorithm [16,19] to compute this concept lattice. We show that the “expert” for the context required by this algorithm can be realized by the subsumption algorithm for \mathcal{L}_2 .

The second problem that we will address is that the choice of the examples is crucial for the quality of the result obtained by the bottom-up construction of concepts. If the examples are too similar, the resulting concept might be too specific. Conversely, if the examples are too different, the resulting concept is likely to be too general. Thus, it would be good to have a tool that supports the process of choosing an appropriate set of objects as examples. Assume that C_1, \dots, C_n are the most specific concepts of a given collection of objects o_1, \dots, o_n , and that we intend to use subsets of this collection for constructing new concepts. In order to avoid obtaining concepts that are too general or too specific, it would be good to know the position of the corresponding lcs in the subsumption hierarchy of all least common subsumers of subsets of $\{C_1, \dots, C_n\}$. Since there are exponentially many subsets to be considered, and (depending on the DL language) both, computing the lcs and testing for subsumption, can be expensive operations, we want to obtain complete information on how this hierarchy looks like without computing the least common subsumers of all subsets of $\{C_1, \dots, C_n\}$, and without explicitly making all the subsumption tests between these least common subsumers. Again, this is where methods of FCA can be utilized. We will define a formal context that has the property that its concept lattice is isomorphic to the *inverse* subsumption hierarchy of all least common subsumers of subsets of $\{C_1, \dots, C_n\}$. The attribute exploration algorithm can again be used to compute this lattice. In fact, the “expert” required by the algorithm can be realized by the subsumption algorithm and the algorithm for computing the lcs.

In the next section, we introduce the relevant notions from description logics, and in Section 3, we introduce as many of the basic notions of formal concept analysis as are necessary for our purposes. In particular, we sketch the attribute exploration algorithm. In Section 4 we show how this algorithm can be used to compute the hierarchy of all conjunctions of concepts defined in a terminology, and in Section 5 we do the same for the hierarchy of all least common subsumers of subsets of a given finite set of concepts. In Section 6, we describe some experimental results. In Section 7 we show that the approaches described in Subsection 4.2 and Section 5 are both instances of a more general approach. Section 8 concludes with some comments on possible future work.

This paper was intended to be an amalgamation of the results originally presented in [1] (see Subsection 4.1) and [10] (see Section 5). However, when reconsidering [1], we saw that in addition to the approach chosen there, one can also use another approach (see Subsection 4.2) to solve the problem addressed in [1]. When comparing this new approach with the approach employed in [10], we saw that both are instances of a more general approach, which is described in

Table 1. Syntax and semantics of concept descriptions and definitions.

name of constructor	Syntax	Semantics	\mathcal{ALC}	\mathcal{EL}
top-concept	\top	$\Delta^{\mathcal{I}}$	x	x
bottom-concept	\perp	\emptyset	x	
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	x	
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	x	x
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	x	
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	x	
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	x	x
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$	x	x

Section 7. The experimental results described in Subsection 6.2 have already been published in [10], whereas the experimental results described in Subsection 6.1 have not been published before.

2 Description Logics

For the purpose of this paper, it is sufficient to restrict the attention to the formalism for defining concepts (i.e., we need not introduce ABoxes, which describe objects and their properties). In order to define concepts in a DL knowledge base, one starts with a set N_C of concept names (unary predicates) and a set N_R of role names (binary predicates), and defines more complex *concept descriptions* using the operations provided by the concept description language of the particular system. In this paper, we consider the DL \mathcal{ALC} and its sublanguage \mathcal{EL}^1 , which allow for concept descriptions built from the indicated subsets of the constructors shown in Table 1. In this table, r stands for a role name, A for a concept name, and C, D for arbitrary concept descriptions. A *concept definition* (as shown in the last row of Table 1) assigns a concept name A to a complex description C . A finite set of such definitions is called a TBox iff it is acyclic (i.e., no definition refers, directly or indirectly, to the name it defines) and unambiguous (i.e., each name has at most one definition).

The semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of \mathcal{I} is a non-empty set and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $P \in N_C$ to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1. The interpretation \mathcal{I} is a model of the TBox \mathcal{T} iff it satisfies all its concept definitions, i.e., $A^{\mathcal{I}} = C^{\mathcal{I}}$ holds for all $A \equiv C$ in \mathcal{T} .

One of the most important traditional inference services provided by DL systems is computing subconcept/superconcept relationships (so-called *subsump-*

¹ It should be noted, however, that the methods developed in this paper in principle apply to arbitrary concept descriptions languages, as long as the more expressive one is equipped with a subsumption algorithm and the less expressive one with an algorithm for computing least common subsumers.

tion relationships) between concept descriptions. The concept description C_2 *subsumes* the concept description C_1 ($C_1 \sqsubseteq C_2$) iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ for all interpretations \mathcal{I} ; C_2 is *equivalent* to C_1 ($C_1 \equiv C_2$) iff $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. The subsumption relation \sqsubseteq is a quasi order (i.e., reflexive and transitive), but in general not a partial order since it need not be antisymmetric (i.e., there may exist equivalent descriptions that are not syntactically equal). As usual, the quasi order \sqsubseteq induces a partial order \sqsubseteq_{\equiv} on the equivalence classes of concept descriptions:

$$[C_1]_{\equiv} \sqsubseteq_{\equiv} [C_2]_{\equiv} \text{ iff } C_1 \sqsubseteq C_2,$$

where $[C_i]_{\equiv} := \{D \mid C_i \equiv D\}$ is the equivalence class of C_i ($i = 1, 2$). When talking about the *subsumption hierarchy* of a set of descriptions, we mean this induced partial order. In the presence of a TBox, subsumption must be computed w.r.t. this TBox. The concept description C_2 *subsumes* the concept description C_1 w.r.t. the TBox \mathcal{T} ($C_1 \sqsubseteq_{\mathcal{T}} C_2$) iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ for all models \mathcal{I} of \mathcal{T} . All the other notions introduced above can be adapted in the obvious way to the case of subsumption w.r.t. a TBox.

Deciding subsumption between \mathcal{EL} -concept descriptions (without or with TBox) is polynomial [9,5], whereas the subsumption problem for \mathcal{ALC} (without or with TBox) is PSPACE-complete [35,26].

In addition to subsumption, we are here interested in the non-standard inference problem of computing the least common subsumer of concept descriptions.

Definition 1 *Given two concept descriptions C_1, C_2 in a DL \mathcal{L} , the concept description C of \mathcal{L} is an lcs of C_1, C_2 in \mathcal{L} ($C = \text{lcs}_{\mathcal{L}}(C_1, C_2)$) iff (i) $C_i \sqsubseteq C$ for $i = 1, 2$, and (ii) C is the least concept description with this property, i.e., if C' satisfies $C_i \sqsubseteq C'$ for $i = 1, 2$, then $C \sqsubseteq C'$.*

The lcs of n concept descriptions is obtained by iterating the application of the binary lcs: $\text{lcs}_{\mathcal{L}}(C_1, \dots, C_n) := \text{lcs}_{\mathcal{L}}(C_1, \dots, \text{lcs}_{\mathcal{L}}(C_{n-1}, C_n) \dots)$. Depending on the DL under consideration, the lcs of two or more descriptions need not always exist, but if it exists, then it is unique up to equivalence. In [9], it is shown that the lcs of two \mathcal{EL} -concept descriptions always exists and that it can be computed in polynomial time; however, the size of the n -ary lcs can be exponential in the size of the input descriptions, and thus computing it may require exponential time.

As an example for the (binary) lcs in \mathcal{EL} , consider the \mathcal{EL} -concept descriptions

$$\begin{aligned} C &:= \exists \text{has-child.}(\text{Male} \sqcap \text{Doctor}) \text{ and} \\ D &:= \exists \text{has-child.}(\text{Male} \sqcap \text{Mechanic}) \sqcap \exists \text{has-child.}(\text{Female} \sqcap \text{Doctor}) \end{aligned}$$

respectively describing parents having a child that is a male doctor, and parents having a son that is a mechanic and a daughter that is a doctor. The lcs of C and D is given by the \mathcal{EL} -concept description

$$\text{lcs}_{\mathcal{EL}}(C, D) = \exists \text{has-child.Male} \sqcap \exists \text{has-child.Doctor}.$$

It describes all parents having at least one son and at least one child that is a doctor. Note that $\text{lcs}_{\mathcal{ALC}}(C, D) = C \sqcup D$.

In order to describe the lcs algorithm for \mathcal{EL} in the general case, we need to introduce some notation. Let C be an \mathcal{EL} -concept description. Then $\text{names}(C)$ denotes the set of concept names occurring in the top-level conjunction of C , $\text{roles}(C)$ the set of role names occurring in an existential restriction on the top-level of C , and $\text{restrict}_r(C)$ denotes the set of all concept descriptions occurring in an existential restriction on the role r on the top-level of C . In the above example, we have $\text{names}(C) = \text{names}(D) = \emptyset$, $\text{restrict}_{\text{has-child}}(C) = \{\text{Male} \sqcap \text{Doctor}\}$, and $\text{restrict}_{\text{has-child}}(D) = \{\text{Male} \sqcap \text{Mechanic}, \text{Female} \sqcap \text{Doctor}\}$.

Now, let C, D be \mathcal{EL} -concept descriptions. Then we have

$$\text{lcs}_{\mathcal{EL}}(C, D) = \bigcap_{A \in \text{names}(C) \cap \text{names}(D)} A \sqcap \bigcap_{r \in \text{roles}(C) \cap \text{roles}(D)} \bigcap_{E \in \text{restrict}_r(C), F \in \text{restrict}_r(D)} \exists r. \text{lcs}_{\mathcal{EL}}(E, F)$$

Here, the empty conjunction stands for the top concept \top . The recursive call of $\text{lcs}_{\mathcal{EL}}$ is well-founded since the role depth (i.e., the maximal nesting of existential restrictions) of the concept descriptions in $\text{restrict}_r(C)$ ($\text{restrict}_r(D)$) is strictly smaller than the role depth of C (D).

If the \mathcal{EL} -concept descriptions C, D contain concept names that are defined in an \mathcal{ALC} -TBox \mathcal{T} , then it is currently not known how to compute their least common subsumer, i.e., the least \mathcal{EL} -concept description containing defined concepts of \mathcal{T} that subsumes C and D w.r.t. \mathcal{T} . If we ignore the TBox by treating all concept names as primitive, and just compute the lcs of C, D , then we obtain a common subsumer of C, D also w.r.t. \mathcal{T} , but it need not be the least one. As a simple example, consider the TBox \mathcal{T} :

$$\begin{aligned} \text{NoSon} &\equiv \forall \text{has-child. Female}, \\ \text{NoDaughter} &\equiv \forall \text{has-child. } \neg \text{Female}, \\ \text{SonRichDoctor} &\equiv \forall \text{has-child. (Female} \sqcup (\text{Doctor} \sqcap \text{Rich})) \\ \text{DaughterHappyDoctor} &\equiv \forall \text{has-child. (} \neg \text{Female} \sqcup (\text{Doctor} \sqcap \text{Happy})) \\ \text{ChildrenDoctor} &\equiv \forall \text{has-child. Doctor} \end{aligned}$$

and the \mathcal{EL} -concept descriptions

$$\begin{aligned} C &:= \exists \text{has-child. (NoSon} \sqcap \text{DaughterHappyDoctor}), \\ D &:= \exists \text{has-child. (NoDaughter} \sqcap \text{SonRichDoctor}). \end{aligned}$$

If we ignore the TBox, then we obtain the \mathcal{EL} -concept description

$$\exists \text{has-child. } \top$$

as common subsumer of C, D . However, if we take into account that both $\text{NoSon} \sqcap \text{DaughterHappyDoctor}$ and $\text{NoDaughter} \sqcap \text{SonRichDoctor}$ are subsumed by ChildrenDoctor , then we obtain the more specific common subsumer

$$\exists \text{has-child. ChildrenDoctor}.$$

This motivates our interest in computing the subsumption hierarchy of all conjunctions of concepts defined in a given TBox. Before we can show how this hierarchy can be computed using tools from FCA, we must introduce the relevant notions from FCA.

3 Formal Concept Analysis

We will introduce only those notions and results from FCA that are necessary for our purposes. Since it is the main FCA tool that we will employ, we will describe how the attribute exploration algorithm works. Note, however, that explaining why it works is beyond the scope of this paper (see [19] for more information on this and FCA in general).

Definition 2 A formal context is a triple $\mathcal{K} = (\mathcal{O}, \mathcal{P}, \mathcal{S})$, where \mathcal{O} is a set of objects, \mathcal{P} is a set of attributes (or properties), and $\mathcal{S} \subseteq \mathcal{O} \times \mathcal{P}$ is a relation that connects each object o with the attributes satisfied by o .

Let $\mathcal{K} = (\mathcal{O}, \mathcal{P}, \mathcal{S})$ be a formal context. For a set of objects $A \subseteq \mathcal{O}$, the *intent* A' of A is the set of attributes that are satisfied by all objects in A , i.e.,

$$A' := \{p \in \mathcal{P} \mid \forall a \in A: (a, p) \in \mathcal{S}\}.$$

Similarly, for a set of attributes $B \subseteq \mathcal{P}$, the *extent* B' of B is the set of objects that satisfy all attributes in B , i.e.,

$$B' := \{o \in \mathcal{O} \mid \forall b \in B: (o, b) \in \mathcal{S}\}.$$

It is easy to see that, for $A_1 \subseteq A_2 \subseteq \mathcal{O}$ (resp. $B_1 \subseteq B_2 \subseteq \mathcal{P}$), we have

- $A_2' \subseteq A_1'$ (resp. $B_2' \subseteq B_1'$),
- $A_1 \subseteq A_1''$ and $A_1' = A_1'''$ (resp. $B_1 \subseteq B_1''$ and $B_1' = B_1'''$).

A *formal concept* is a pair (A, B) consisting of an extent $A \subseteq \mathcal{O}$ and an intent $B \subseteq \mathcal{P}$ such that $A' = B$ and $B' = A$. Such formal concepts can be hierarchically ordered by inclusion of their extents, and this order (denoted by \leq in the following) induces a complete lattice, the *concept lattice* of the context. Given a formal context, the first step for analyzing this context is usually to compute the concept lattice.

The following are easy consequences of the definition of formal concepts and the properties of the \cdot' operation mentioned above:

Lemma 3 All formal concepts are of the form (A'', A') for a subset A of \mathcal{O} , and any such pair is a formal concept. In addition, $(A_1'', A_1') \leq (A_2'', A_2')$ iff $A_2' \subseteq A_1'$.

Thus, if the context is finite, the concept lattices can in principle be computed by enumerating the subsets A of \mathcal{O} , and applying the operations \cdot' and \cdot'' . However, this naïve algorithm is usually very inefficient. In many applications [37], one has a large (or even infinite) set of objects, but only a relatively small

set of attributes. In such a situation, Ganter's *attribute exploration* algorithm [16,19] has turned out to be an efficient approach for computing the concept lattice.

In one of the applications considered in this paper, we are faced with the *dual* situation: the set of attributes is the *infinite* set of all possible concept descriptions of the DL under consideration, and the set of objects is the *finite* collection of concept descriptions for which we want to compute the subsumption hierarchy of least common subsumers. To overcome this problem, one can either dualize the attribute exploration algorithm and the notions on which it depends, as done in [10]. In this paper, we follow an alternative approach: we considered the dual context (which is obtained by transposing the matrix corresponding to \mathcal{S}) and employed the usual attribute exploration for this context. The concept lattice of the dual context is obviously the dual of the concept lattice of the original context, i.e., the corresponding orderings on the formal concepts are inverses of each other.

Attribute Exploration

Before we can describe the attribute exploration algorithm, we must introduce some notation. The most important notion for the algorithm is the one of an implication between sets of attributes. Intuitively, such an implication $B_1 \rightarrow B_2$ holds if any object satisfying all elements of B_1 also satisfies all elements of B_2 .

Definition 4 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P}, \mathcal{S})$ be a formal context and B_1, B_2 be subsets of \mathcal{P} . The implication $B_1 \rightarrow B_2$ holds in \mathcal{K} ($\mathcal{K} \models B_1 \rightarrow B_2$) iff $B'_1 \subseteq B'_2$. An object o violates the implication $B_1 \rightarrow B_2$ iff $o \in B'_1 \setminus B'_2$.

It is easy to see that an implication $B_1 \rightarrow B_2$ holds in \mathcal{K} iff $B_2 \subseteq B'_1$. In particular, given a set of attributes B , the implications $B \rightarrow B''$ and $B \rightarrow (B'' \setminus B)$ always hold in \mathcal{K} . We denote the set of all implications that hold in \mathcal{K} by $\text{Imp}(\mathcal{K})$. This set can be very large, and thus one is interested in (small) generating sets.

Definition 5 Let \mathcal{J} be a set of implications, i.e., the elements of \mathcal{J} are of the form $B_1 \rightarrow B_2$ for sets of attributes $B_1, B_2 \subseteq \mathcal{P}$. For a subset B of \mathcal{P} , the implication hull of B with respect to \mathcal{J} is denoted by $\mathcal{J}(B)$. It is the smallest subset H of \mathcal{P} such that

- $B \subseteq H$, and
- $B_1 \rightarrow B_2 \in \mathcal{J}$ and $B_1 \subseteq H$ imply $B_2 \subseteq H$.

The set of implications generated by \mathcal{J} consists of all implications $B_1 \rightarrow B_2$ such that $B_2 \subseteq \mathcal{J}(B_1)$. It will be denoted by $\text{Cons}(\mathcal{J})$. We say that a set of implications \mathcal{J} is a base of $\text{Imp}(\mathcal{K})$ iff $\text{Cons}(\mathcal{J}) = \text{Imp}(\mathcal{K})$ and no proper subset of \mathcal{J} satisfies this property.

If \mathcal{J} is a base for $\text{Imp}(\mathcal{K})$, then it can be shown that $B'' = \mathcal{J}(B)$ for all $B \subseteq \mathcal{P}$. The implication hull $\mathcal{J}(B)$ of a set of attributes B can be computed in time linear

in the size of \mathcal{J} and B using, for example, methods for deciding satisfiability of sets of propositional Horn clauses [13]. Consequently, given a base \mathcal{J} for $\text{Imp}(\mathcal{K})$, any question of the form “ $B_1 \rightarrow B_2 \in \text{Imp}(\mathcal{K})$?” can be answered in time linear in the size of $\mathcal{J} \cup \{B_1 \rightarrow B_2\}$.

There may exist different implication bases of $\text{Imp}(\mathcal{K})$, and not all of them need to be of minimal cardinality. A base \mathcal{J} of $\text{Imp}(\mathcal{K})$ is called *minimal base* iff no base of $\text{Imp}(\mathcal{K})$ has a cardinality smaller than the cardinality of \mathcal{J} . Duquenne and Guigues have given a description of such a minimal base [14]. Ganter’s attribute exploration algorithm computes this minimal base as a by-product. In the following, we define the Duquenne-Guigues base and show how it can be computed using the attribute exploration algorithm.

The definition of the Duquenne-Guigues base given below is based on a modification of the closure operator $B \mapsto \mathcal{J}(B)$ defined by a set \mathcal{J} of implications. For a subset B of \mathcal{P} , the *implication pseudo-hull* of B with respect to \mathcal{J} is denoted by $\mathcal{J}^*(B)$. It is the smallest subset H of \mathcal{P} such that

- $B \subseteq H$, and
- $B_1 \rightarrow B_2 \in \mathcal{J}$ and $B_1 \subset H$ (strict subset) imply $B_2 \subseteq H$.

Given \mathcal{J} , the pseudo-hull of a set $B \subseteq \mathcal{P}$ can again be computed in time linear in the size of \mathcal{J} and B (e.g., by adapting the algorithm in [13] appropriately). A subset B of \mathcal{P} is called *pseudo-closed* in a formal context \mathcal{K} iff $\text{Imp}(\mathcal{K})^*(B) = B$ and $B'' \neq B$.

Definition 6 *The Duquenne-Guigues base of a formal context \mathcal{K} consists of all implications $B_1 \rightarrow B_2$ where $B_1 \subseteq \mathcal{P}$ is pseudo-closed in \mathcal{K} and $B_2 = B_1'' \setminus B_1$.*

When trying to use this definition for actually *computing* the dual Duquenne-Guigues base of a formal context, one encounters two problems:

1. The definition of pseudo-closed refers to the set of all valid implications $\text{Imp}(\mathcal{K})$, and our goal is to avoid explicitly computing all of them.
2. The closure operator $B \mapsto B''$ is used, and computing it via $B \mapsto B' \mapsto B''$ may not be feasible for a context with a larger or infinite set of objects.

Ganter solves the first problem by enumerating the pseudo-closed sets of \mathcal{K} in a particular order, called *lectic order*. This order makes sure that it is sufficient to use the already computed part \mathcal{J} of the base when computing the pseudo-hull. To define the lectic order, fix an arbitrary linear order on the set of attributes $\mathcal{P} = \{p_1, \dots, p_n\}$, say $p_1 < \dots < p_n$. For all $j, 1 \leq j \leq n$, and $B_1, B_2 \subseteq \mathcal{P}$ we define

$$B_1 <_j B_2 \text{ iff } p_j \in B_2 \setminus B_1 \text{ and } B_1 \cap \{p_1, \dots, p_{j-1}\} = B_2 \cap \{p_1, \dots, p_{j-1}\}.$$

The lectic order $<$ is the union of all relations $<_j$ for $j = 1, \dots, n$. It is a linear order on the powerset of \mathcal{P} . The lectic smallest subset of \mathcal{P} is the empty set.

The second problem is solved by constructing an increasing chain of finite subcontexts of \mathcal{K} . The context $\mathcal{K}_i = (\mathcal{O}_i, \mathcal{P}_i, \mathcal{S}_i)$ is a *subcontext* of \mathcal{K} iff $\mathcal{O}_i \subseteq \mathcal{O}$,

$\mathcal{P}_i = \mathcal{P}$, and $\mathcal{S}_i = \mathcal{S} \cap (\mathcal{O}_i \times \mathcal{P})$. The closure operator $B \mapsto B''$ is always computed with respect to the current finite subcontext \mathcal{K}_i . To avoid adding a wrong implication, an “expert” is asked whether the implication $B \rightarrow B'' \setminus B$ really holds in the whole context \mathcal{K} . If it does not hold, the expert must provide a counterexample, i.e., an object o from $\mathcal{O} \setminus \mathcal{O}_i$ that violates the implication. This object is then added to the current context. Technically, this means that the expert must provide an object o , and must say which of the attributes in \mathcal{P} are satisfied for this object.

The following algorithm computes the set of all intents of formal concepts of \mathcal{K} as well as the Duquenne-Guigues base of \mathcal{K} . The concept lattice is then given by the inverse inclusion ordering between the intents.

Algorithm 7 (Attribute exploration)

Initialization: One starts with the empty set of implications, i.e., $\mathcal{J}_0 := \emptyset$, the empty set of concept intents $\mathcal{C}_0 := \emptyset$, and the empty subcontext \mathcal{K}_0 of \mathcal{K} , i.e., $\mathcal{O}_0 := \emptyset$. The lectic smallest subset of \mathcal{P} is $B_0 := \emptyset$.

Iteration: Assume that \mathcal{K}_i , \mathcal{J}_i , \mathcal{C}_i , and B_i ($i \geq 0$) are already computed. Compute B_i'' with respect to the current subcontext \mathcal{K}_i . Now the expert is asked whether the implication $B_i \rightarrow B_i'' \setminus B_i$ holds in \mathcal{K} ².

If the answer is “no”, then let $o_i \in \mathcal{O}$ be the counterexample provided by the expert. Let $B_{i+1} := B_i$, $\mathcal{J}_{i+1} := \mathcal{J}_i$, and let \mathcal{K}_{i+1} be the subcontext of \mathcal{K} with $\mathcal{O}_{i+1} := \mathcal{O}_i \cup \{o_i\}$. The iteration continues with \mathcal{K}_{i+1} , \mathcal{J}_{i+1} , \mathcal{C}_{i+1} , and B_{i+1} .

If the answer is “yes”, then $\mathcal{K}_{i+1} := \mathcal{K}_i$ and

$$(\mathcal{C}_{i+1}, \mathcal{J}_{i+1}) := \begin{cases} (\mathcal{C}_i, \mathcal{J}_i \cup \{B_i \rightarrow B_i'' \setminus B_i\}) & \text{if } B_i'' \neq B_i, \\ (\mathcal{C}_i \cup \{B_i\}, \mathcal{J}_i) & \text{if } B_i'' = B_i. \end{cases}$$

To find the new set B_{i+1} , we start with $j = n$, and test whether

$$(*) \quad B_i <_j \mathcal{J}_{i+1}^*((B_i \cap \{p_1, \dots, p_{j-1}\}) \cup \{p_j\})$$

holds. The index j is decreased until one of the following cases occurs:

(1) $j = 0$: In this case, \mathcal{C}_{i+1} is the set of all concept intents and \mathcal{J}_{i+1} the Duquenne-Guigues base of \mathcal{K} , and the algorithm stops.

(2) $(*)$ holds for $j > 0$: In this case, $B_{i+1} := \mathcal{J}_{i+1}^*((B_i \cap \{p_1, \dots, p_{j-1}\}) \cup \{p_j\})$, and the iteration is continued.

4 Subsumption between Conjunctions of Concepts

Most of the results that we will show are independent of the DL under consideration. The only restriction that we need is that it allows for conjunction.

In the following, let \mathcal{T} be a fixed TBox, and let p_1, \dots, p_n be the concepts defined in \mathcal{T} . We are interested in representing all subsumption relationships (w.r.t. \mathcal{T}) between finite conjunctions of these defined concepts. In order to

² If $B_i'' \setminus B_i = \emptyset$, then it is not really necessary to ask the expert because implications with empty right-hand side hold in any context.

employ tools from FCA for this purpose, we want to define a formal context such that the concept lattice of this context is isomorphic to the subsumption hierarchy we are interested in. In this section, we will describe two possible ways for defining such a context. The first one (which was first described in [1]) uses interpretations as objects, whereas the second one uses concepts as objects. The advantage of the second approach is that one can use an arbitrary subsumption algorithm for the DL under consideration as expert. Thus, one can, for example, use highly optimized DL systems like FaCT and RACER for this purpose. In contrast, the first approach requires an extended subsumption algorithm. Though this extended algorithm exists for most of the standard DLs (in particular, for \mathcal{ALC}), one cannot use standard implementations.

4.1 The Semantic Context

The idea underlying the following definition is that a counterexample to a subsumption relationship $C \sqsubseteq_{\mathcal{T}} D$ is a model \mathcal{I} of \mathcal{T} together with an element $d \in \Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}} \setminus D^{\mathcal{I}}$. The objects of the context are all possible such counterexamples.

Definition 8 *The context $\mathcal{K}_{\mathcal{T}} = (\mathcal{O}, \mathcal{P}, \mathcal{S})$ is defined as follows:*

$$\begin{aligned}\mathcal{O} &:= \{(\mathcal{I}, d) \mid \mathcal{I} \text{ is a model of } \mathcal{T} \text{ and } d \in \Delta^{\mathcal{I}}\}, \\ \mathcal{P} &:= \{p_1, \dots, p_n\}, \\ \mathcal{S} &:= \{((\mathcal{I}, d), p) \mid d \in p^{\mathcal{I}}\}.\end{aligned}$$

For a nonempty subset $B = \{p_{i_1}, \dots, p_{i_r}\}$ of \mathcal{P} , we denote the conjunction $p_{i_1} \sqcap \dots \sqcap p_{i_r}$ by $\sqcap B$. For the empty set, we define $\sqcap \emptyset := \top$.

Lemma 9 *Let B_1, B_2 be subsets of \mathcal{P} . The implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\mathcal{T}}$ iff $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$.*

Proof. Assume that $B_1 \rightarrow B_2$ does not hold in $\mathcal{K}_{\mathcal{T}}$. This is the case iff there exists an object $(\mathcal{I}, d) \in B_1' \setminus B_2'$. By definition of \mathcal{S} and of the operator $B \mapsto B'$, this means that (1) $d \in p^{\mathcal{I}}$ for all $p \in B_1$, and (2) there exists $p' \in B_2$ such that $d \notin p'^{\mathcal{I}}$. By the semantics of the conjunction operator, (1) is equivalent to $d \in (\sqcap B_1)^{\mathcal{I}}$, and (2) is equivalent to $d \notin (\sqcap B_2)^{\mathcal{I}}$. Since \mathcal{I} is a model of \mathcal{T} , this shows that the subsumption relationship $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$ does not hold. Obviously, all of the conclusions we have made are reversible. \square

Thus, the Duquenne-Guigues base \mathcal{L} of $\mathcal{K}_{\mathcal{T}}$ also yields a representation of all subsumption relationships of the form $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$ for subsets B_1, B_2 of \mathcal{P} . As mentioned in Section 3, any question “ $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$?” can then be answered in time linear in the size of $\mathcal{L} \cup \{B_1 \rightarrow B_2\}$.

Theorem 10 *The concept lattice of the context $\mathcal{K}_{\mathcal{T}}$ is isomorphic to the subsumption hierarchy of all conjunctions of subsets of \mathcal{P} w.r.t. \mathcal{T} .*

Proof. In order to obtain an appropriate isomorphism, we define a mapping π from the formal concepts of the context $\mathcal{K}_{\mathcal{T}}$ to the set of all (equivalence classes of) conjunctions of subsets of \mathcal{P} as follows:

$$\pi(A, B) = [\sqcap B]_{\equiv}.$$

For formal concepts $(A_1, B_1), (A_2, B_2)$ of $\mathcal{K}_{\mathcal{T}}$ we have $(A_1, B_1) \leq (A_2, B_2)$ iff $B_2 \subseteq B_1$. Since B_1 is the intent of the formal concept (A_1, B_1) , we have $B_1 = A'_1 = A''_1 = B'_1$, and thus $B_2 \subseteq B_1$ iff $B_2 \subseteq B'_1$ iff the implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\mathcal{T}}$ iff $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$. Overall, we have thus shown that π is an order embedding (and thus injective): $(A_1, B_1) \leq (A_2, B_2)$ iff $[\sqcap B_1]_{\equiv} \sqsubseteq_{\equiv} [\sqcap B_2]_{\equiv}$.

It remains to be shown that π is surjective as well. Let B be an arbitrary subset of \mathcal{P} . We must show that $[\sqcap B]_{\equiv}$ can be obtained as an image under the mapping π . We know that (B', B'') is a formal concept of $\mathcal{K}_{\mathcal{T}}$, and thus it is sufficient to show that $\pi(B', B'') = [\sqcap B]_{\equiv}$, i.e., $\sqcap(B'') \equiv_{\mathcal{T}} \sqcap B$. Obviously, $B \subseteq B''$ implies $\sqcap(B'') \sqsubseteq_{\mathcal{T}} \sqcap B$. Conversely, the implication $B \rightarrow B''$ holds in $\mathcal{K}_{\mathcal{T}}$, and thus Lemma 9 yields $\sqcap B \sqsubseteq_{\mathcal{T}} \sqcap B''$. \square

If we want to apply Algorithm 7 to compute the concept lattice and the Duquenne-Guigues base, we need an “expert” for the context $\mathcal{K}_{\mathcal{T}}$. This expert must be able to answer the questions asked by the attribute exploration algorithm, i.e., given an implication $B_1 \rightarrow B_2$, it must be able to decide whether this implication holds in $\mathcal{K}_{\mathcal{T}}$. If the implication does not hold, it must be able to compute a counterexample, i.e., an object $o \in B'_1 \setminus B'_2$.

By Lemma 9, $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\mathcal{T}}$ iff $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$. Thus, validity of implications in $\mathcal{K}_{\mathcal{T}}$ can be decided using a standard subsumption algorithm. A counterexample to the implication $B_1 \rightarrow B_2$ is a pair $(d, \mathcal{I}) \in \mathcal{O}$ such that $d \in (\sqcap B_1)^{\mathcal{I}} \setminus (\sqcap B_2)^{\mathcal{I}}$. Since the usual tableau-based subsumption algorithms [11] in principle try to generate finite countermodels to subsumption relationships, they can usually be extended such that they yield such an object in case the subsumption relationship does not hold. In [1], this is explicitly shown for the DL \mathcal{ALC} .

Proposition 11 *Let \mathcal{T} be an \mathcal{ALC} TBox. The tableau-based subsumption algorithm for \mathcal{ALC} can be extended such that it functions as an “expert” for the context $\mathcal{K}_{\mathcal{T}}$ without increasing its worst-case complexity of PSPACE.*

However, the highly optimized algorithms in systems like FaCT and RACER do not produce such countermodels as output. For this reason, we are interested in a context that has the same attributes and the same concept lattice (up to isomorphism), but for which a standard subsumption algorithm can function as an expert.

4.2 The Syntactic Context

The intuition underlying this context is that the subsumption relationship $C \sqsubseteq_{\mathcal{T}} D$ does not hold iff there is a concept E such that $E \sqsubseteq_{\mathcal{T}} C$ and $E \not\sqsubseteq_{\mathcal{T}} D$.

Definition 12 The context $\mathcal{K}'_{\mathcal{T}} = (\mathcal{O}', \mathcal{P}, \mathcal{S}')$ is defined as follows:

$$\begin{aligned}\mathcal{O}' &:= \{E \mid E \text{ is a concept description of the DL under consideration}\}; \\ \mathcal{P} &:= \{p_1, \dots, p_n\}, \\ \mathcal{S}' &:= \{(E, p) \mid E \sqsubseteq_{\mathcal{T}} p\}.\end{aligned}$$

The context $\mathcal{K}'_{\mathcal{T}}$ satisfies the analogs of Lemma 9 and Theorem 10.

Lemma 13 Let B_1, B_2 be subsets of \mathcal{P} . The implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}'_{\mathcal{T}}$ iff $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$.

Proof. First, we prove the *only if* direction. If the implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}'_{\mathcal{T}}$, then this means that the following holds for all objects $E \in \mathcal{O}'$: if $E \sqsubseteq_{\mathcal{T}} p$ holds for all $p \in B_1$, then $E \sqsubseteq_{\mathcal{T}} p$ also holds for all $p \in B_2$. Thus, if we take $\sqcap B_1$ as object E , we obviously have $\sqcap B_1 \sqsubseteq_{\mathcal{T}} p$ for all $p \in B_1$, and thus $\sqcap B_1 \sqsubseteq_{\mathcal{T}} p$ for all $p \in B_2$, which shows $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$.

Second, we prove the *if* direction. If $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$, then any object E satisfying $E \sqsubseteq_{\mathcal{T}} \sqcap B_1$ also satisfies $E \sqsubseteq_{\mathcal{T}} \sqcap B_2$ by the transitivity of the subsumption relation. Consequently, if E is a subconcept of all concepts in B_1 , then it is also a subconcept of all concepts in B_2 , i.e., if E satisfies all attributes in B_1 , it also satisfies all attributes in B_2 . This shows that the implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}'_{\mathcal{T}}$. \square

The proof of the following theorem is identical to the proof of Theorem 10.

Theorem 14 The concept lattice of the context $\mathcal{K}'_{\mathcal{T}}$ is isomorphic to the subsumption hierarchy of all conjunctions of subsets of \mathcal{P} w.r.t. \mathcal{T} .

Again, attribute exploration can be used to compute the concept lattice since any standard subsumption algorithm for the DL under consideration can be used as an expert for $\mathcal{K}'_{\mathcal{T}}$.

Proposition 15 Any decision procedure for subsumption functions as an expert for the context $\mathcal{K}'_{\mathcal{T}}$.

Proof. The attribute exploration algorithm asks questions of the form “ $B_1 \rightarrow B_2$?” By Lemma 13, we can translate these questions into subsumption questions of the form “ $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2$?” Obviously, any decision procedure for subsumption can answer these questions correctly.

Now, assume that $B_1 \rightarrow B_2$ does *not* hold in $\mathcal{K}'_{\mathcal{T}}$, i.e., $\sqcap B_1 \not\sqsubseteq_{\mathcal{T}} \sqcap B_2$. We claim that $\sqcap B_1$ is a counterexample, i.e., $\sqcap B_1 \in B'_1$, but $\sqcap B_1 \notin B'_2$. This is an immediate consequence of the facts that $B'_i = \{E \mid E \sqsubseteq_{\mathcal{T}} \sqcap B_i\}$ ($i = 1, 2$) and that $\sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_1$ and $\sqcap B_1 \not\sqsubseteq_{\mathcal{T}} \sqcap B_2$. \square

5 Computing the Hierarchy of Least Common Subsumers

In the following, we assume that in the DL \mathcal{L} under consideration the lcs always exists and can effectively be computed.

Given a finite set $\mathcal{C} := \{C_1, \dots, C_n\}$ of concept descriptions, we are interested in the subsumption hierarchy between all least common subsumers of subsets of \mathcal{C} . For sets $B \subseteq \mathcal{C}$ of cardinality ≥ 2 , we have already defined the notion $\text{lcs}(B)$. We extend this notion to the empty set and singletons in the obvious way: $\text{lcs}(\emptyset) := \perp$ and $\text{lcs}(\{C_i\}) := C_i$.

Our goal is to compute the subsumption hierarchy between all concept descriptions $\text{lcs}(B)$ for subsets B of \mathcal{C} without explicitly computing all these least common subsumers. This is again achieved by defining a formal context (with attribute set \mathcal{C}) such that the concept lattice of this context is isomorphic to the subsumption hierarchy we are interested in. The following context is similar to the syntactic context defined in the previous section. The main difference is the definition of the incidence relation, where subsumption is used in the opposite direction.

Definition 16 *Given a DL language \mathcal{L} and a finite set $\mathcal{C} := \{C_1, \dots, C_n\}$ of \mathcal{L} -concept descriptions, the corresponding formal context $\mathcal{K}_{\mathcal{L}}(\mathcal{C}) = (\mathcal{O}'', \mathcal{P}'', \mathcal{S}'')$ is defined as follows:*

$$\begin{aligned}\mathcal{O}'' &:= \{D \mid D \text{ is an } \mathcal{L}\text{-concept description}\}, \\ \mathcal{P}'' &:= \mathcal{C}, \\ \mathcal{S}'' &:= \{(D, C) \mid C \sqsubseteq D\}.\end{aligned}$$

As an easy consequence of the definition of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ and of the lcs , we obtain that the extent of a set $B \subseteq \mathcal{P}''$ is closely related to the lcs of this set:

Lemma 17 *Let B, B_1, B_2 be subsets of \mathcal{P}'' .*

1. $B' = \{D \in \mathcal{O}'' \mid \text{lcs}(B) \sqsubseteq D\}$.
2. $B'_1 \subseteq B'_2$ iff $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$.

Proof. First, let B be a subset of \mathcal{P}'' . We have $D \in B'$ iff $C \sqsubseteq D$ for all $C \in B$ iff $\text{lcs}(B) \sqsubseteq D$.

Second, by 1. we have $B'_1 \subseteq B'_2$ iff $\text{lcs}(B_1) \sqsubseteq D$ implies $\text{lcs}(B_2) \sqsubseteq D$ for all $D \in \mathcal{O}''$. Thus, if $B'_1 \subseteq B'_2$, then $\text{lcs}(B_1) \sqsubseteq \text{lcs}(B_1)$ yields $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$. Conversely, if $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$, then $\text{lcs}(B_1) \sqsubseteq D$ obviously implies $\text{lcs}(B_2) \sqsubseteq D$ for arbitrary concepts D . \square

Now, we can again show that implications correspond to subsumption relationships between the corresponding least common subsumers.

Lemma 18 *Let B_1, B_2 be subsets of \mathcal{P}'' . The implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ iff $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$.*

Proof. $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$
iff $B'_1 \subseteq B'_2$ (by definition)
iff $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$ (by 2. of the above lemma). \square

As an immediate consequence of this lemma, the Duquenne-Guigues base \mathcal{J} of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ yields a representation of all subsumption relationships of the form

$\text{lcs}(B_1) \sqsubseteq \text{lcs}(B_2)$ for subsets B_1, B_2 of \mathcal{O} . Given this base \mathcal{J} , any question of the form “ $\text{lcs}(B_1) \sqsubseteq \text{lcs}(B_2)$?” can then be answered in time linear in the size of $\mathcal{J} \cup \{B_1 \rightarrow B_2\}$. Another easy consequence of the lemma is that the concept lattice of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ coincides with the inverse subsumption hierarchy of all least common subsumers of subsets of \mathcal{C} . The proof of this fact (which we include for the sake of completeness) is very similar to the proof of Theorem 10.

Theorem 19 *The concept lattice of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ is isomorphic to the inverse subsumption hierarchy of all least common subsumers of subsets of \mathcal{C} .*

Proof. We define the mapping π from the formal concepts of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ to the set of (equivalence classes of) least common subsumers of subsets of \mathcal{C} as follows:

$$\pi(A, B) := [\text{lcs}(B)]_{\equiv}.$$

For formal concepts $(A_1, B_1), (A_2, B_2)$ we have $(A_1, B_1) \leq (A_2, B_2)$ iff $A_1 = B'_1 \sqsubseteq A_2 = B'_2$ iff $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$. As an easy consequence we obtain that π is an order embedding (and thus also injective):

$$(A_1, B_1) \leq (A_2, B_2) \text{ iff } [\text{lcs}(B_1)]_{\equiv} \sqsupseteq [\text{lcs}(B_2)]_{\equiv}.$$

It remains to be shown that π is surjective as well. Let B be an arbitrary subset of $\mathcal{C} = \mathcal{P}''$. We must show that $[\text{lcs}(B)]_{\equiv}$ can be obtained as an image under the mapping π . By the dual of Lemma 3, (B', B'') is a formal concept, and thus it is sufficient to show that $\text{lcs}(B) \equiv \text{lcs}(B'')$. Obviously, $B \subseteq B''$ implies $\text{lcs}(B) \sqsubseteq \text{lcs}(B'')$ (by definition of the lcs). Conversely, the implication $B \rightarrow B''$ holds in $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$, and thus $\text{lcs}(B'') \sqsubseteq \text{lcs}(B)$ (by Lemma 18). \square

If we want to apply Algorithm 7 to compute the concept lattice and the Duquenne-Guigues base, we need an “expert” for the context $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$. This expert must be able to answer the questions asked by the attribute exploration algorithm, i.e., given an implication $B_1 \rightarrow B_2$, it must be able to decide whether this implication holds in $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$. If the implication does not hold, it must be able to compute a counterexample, i.e., an object $o \in B'_1 \setminus B'_2$.

If the language \mathcal{L} is such that the lcs is computable and subsumption is decidable (which is, e.g., the case for $\mathcal{L} = \mathcal{EL}$), then we can implement such an expert.

Proposition 20 *Given a subsumption algorithm for \mathcal{L} as well as an algorithm for computing the lcs of a finite set of \mathcal{L} -concept descriptions, these algorithms can be used to obtain an expert for the context $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$.*

Proof. First, we show how to decide whether a given implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ or not. By Lemma 18, we know that $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ iff $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$. Obviously, $\text{lcs}(B_2) \sqsubseteq \text{lcs}(B_1)$ iff $C_i \sqsubseteq \text{lcs}(B_1)$ for all $C_i \in B_2$. Thus, to answer the question “ $B_1 \rightarrow B_2$?”, we first compute $\text{lcs}(B_1)$ and then use the subsumption algorithm to test whether $C_i \sqsubseteq \text{lcs}(B_1)$ holds for all $C_i \in B_2$.

Second, assume that $B_1 \rightarrow B_2$ does not hold in $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$, i.e., $\text{lcs}(B_2) \not\sqsubseteq \text{lcs}(B_1)$. We claim that $\text{lcs}(B_1)$ is a counterexample, i.e., $\text{lcs}(B_1) \in B'_1$ and $\text{lcs}(B_1) \notin B'_2$.

This is an immediate consequence of the facts that $B'_i = \{D \in \mathcal{O}'' \mid \text{lcs}(B_i) \sqsubseteq D\}$ ($i = 1, 2$) and that $\text{lcs}(B_1) \sqsubseteq \text{lcs}(B_1)$ and $\text{lcs}(B_2) \not\sqsubseteq \text{lcs}(B_1)$.

Of this counterexample, Algorithm 7 really needs the row corresponding to this object in the matrix corresponding to \mathcal{S}'' . This row can easily be computed using the subsumption algorithm: for each $C_i \in \mathcal{C} = \mathcal{P}''$, we use the subsumption algorithm to test whether $C_i \sqsubseteq \text{lcs}(B_1)$ holds or not. \square

Using this expert, an application of Algorithm 7 yields

- all intents of formal concepts of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$, and thus the concept lattice of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$, which coincides with the inverse subsumption hierarchy of all least common subsumers of subsets of \mathcal{C} (by Theorem 19);
- the Duquenne-Guigues base of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$, which yields a compact representation of this hierarchy (by Lemma 18); and
- a finite subcontext of $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ that has the same concept intents as $\mathcal{K}_{\mathcal{L}}(\mathcal{C})$ and the same \cdot'' operation on sets of attributes.

Using the output of Algorithm 7, one can then employ the usual tools for drawing concept lattices [38] in order to present the subsumption hierarchy of all least common subsumers of subsets of \mathcal{C} to the knowledge engineer.

6 Some Experimental Results

In the previous two sections, we have shown that the attribute exploration algorithm can be used to compute the hierarchy of least common subsumers of a given set of concept descriptions and the hierarchy of all conjunctions of concepts defined in a terminology. What remains is to analyze whether attribute exploration really is a good approach for solving this task. Our reason for trying it in the first place was that computing this hierarchy is the same as computing a certain concept lattice (as shown above), and that attribute exploration is known to be a very good method for doing this. The problem with this generic argument in favor of attribute exploration is, of course, that we consider a very specific context, and that it might well be that, for this context, attribute exploration is not the best thing to do.

6.1 Results for Computing the Hierarchy of Conjunctions of Defined Concepts

In [29], the approach for computing this hierarchy based on the semantic context (see Subsection 4.1) was implemented and then evaluated on randomly generated \mathcal{ALC} TBoxes. For each TBox size (where size is the number of defined concepts), at least 50 different TBoxes were generated and attribute exploration was applied to the semantic context induced by these TBoxes. The main observations made during these experiments are the following:

1. For TBoxes of size ≥ 30 , the computation of the full hierarchy in all cases took longer than the time out of 5 minutes imposed on each experiment.

2. The size of the Duquenne-Guigues base was very small compared with the number of computed counterexamples and the number of formal concepts of the context. For example, for TBoxes of size 20, there was an average of 13 implications in the base, 850 counterexamples, and 13500 formal concepts.
3. The overhead of the “concept analysis part” of the algorithm was considerable. Although calls to the expert are quite expensive (since the expert is realized by a PSPACE algorithm, the extended subsumption algorithm) less than 60% of the time was spent by the expert.
4. The time required by the extended subsumption algorithm (computing a counterexample) was compared with the time required by a normal subsumption algorithm (answering only “yes” or “no” to subsumption queries). It turned out that for all TBox sizes the runtime of the extended subsumption algorithm was 2.5 times the runtime of the normal subsumption algorithm.
5. The subsumption hierarchy of all conjunctions of concepts defined in a TBox can, of course, also be computed by extending the TBox by a new definition for each such conjunction. However, even if one employs the sophisticated optimization techniques described in [7] to compute the subsumption hierarchy of this extended TBox, the number of calls to the subsumption algorithm is much larger than the number of expert calls during attribute exploration of the semantic context of the unextended TBox. For example, for TBoxes of size 10, attribute exploration required an average of 75 calls of the expert, whereas computing the subsumption hierarchy of the extended TBoxes required 19700 calls of the normal subsumption algorithm.

A few comments regarding these results are in order. First, there are two reasons why TBoxes with more than 30 defined concepts could not be handled in reasonable time. On the one hand, the extended subsumption algorithm was implemented in a naive and almost unoptimized way, and thus could not handle the large concept descriptions (obtained by unfolding the TBox definitions) in an efficient way. On the other hand, for large TBoxes, the size of the concept lattice was huge (and thus a lot of time was spent in the “concept analysis part” of the algorithm). The reason for the huge number of formal concepts compared to the number of implications is probably that the dependencies between the different attributes was quite low in the semantic contexts induced by the randomly generated TBoxes. By varying the probability of using an already defined concept in the definition of a new concept, we were able to generate TBoxes with more or less dependencies between the attributes. However, even with high dependency between attributes, the concept lattice was quite large. As long as this is the case, the “concept analysis part” of the algorithm will require a considerable amount of time.

The constant factor of 2.5 between the runtime of the expert and the runtime of the normal subsumption algorithm is consistent with the theoretical result that both algorithms belong to the same complexity class (PSPACE). For practical purposes, the situation is, however, worse than indicated by this rather small factor. The algorithms we tested were both naive, almost unoptimized implementations. For the normal subsumption algorithm there are now highly

optimized implementations that are several orders of magnitude better than the naive implementation, whereas there is no such optimized implementation available for the extended algorithm. For this reason, using the syntactic context (which just requires a normal subsumption algorithm as expert) appears to be the better option (though this must still be verified in experiments).

In spite of the overhead caused by the expert and the “concept analysis part” of the algorithm, attribute exploration is much better than the naive approach of extending the TBox by a new definition for each conjunction of defined concepts. Thus, the generic argument in favor of attribute exploration is indeed supported by our experiments. As we will see in the next subsection, the same is true for the problem of computing the hierarchy of least common subsumers.

6.2 Results for Computing the Hierarchy of Least Common Subsumers

We have used the bottom-up construction of knowledge bases in a chemical process engineering application [27,34,28], where the knowledge base describes standard building blocks of process models (such as certain types of reactors). When we performed the experiments, this knowledge base consisted of about 600 definitions of building blocks.

In order to test the attribute exploration algorithm, we have taken 7 descriptions of reactors of a similar type, which the process engineers considered to be good examples for generating a new concept. These descriptions were translated into concept descriptions R_1, \dots, R_7 , and we applied the attribute exploration algorithm to this set of attributes. The resulting hierarchy of all least common subsumers of subsets of $\mathcal{C} := \{R_1, \dots, R_7\}$ is depicted in Figure 1. The concept on the top corresponds to the lcs obtained from the whole set of examples, and the concept at the bottom is the lcs obtained from the empty set, i.e., the description \perp . The node labeled $\text{lcs}(i_1 \dots i_m)$ corresponds to the formal concept with intent R_{i_1}, \dots, R_{i_m} , and thus to $\text{lcs}(R_{i_1}, \dots, R_{i_m})$. Note that in many cases $\text{lcs}(R_{i_1}, \dots, R_{i_m})$ can also be obtained as the lcs of a strict subset of $\{R_{i_1}, \dots, R_{i_m}\}$. This can be easily seen by using the least upper-bound operation of the (inverse) concept lattice. For example, $\text{lcs}(R_i, R_7) = \text{lcs}(R_1, \dots, R_7)$ for all $i, 1 \leq i \leq 6$.

Statistical Information: The Duquennes-Guigues base of the context consists of 15 implications, and the concept lattice of 30 formal concepts. If we subtract the trivial least common subsumers \perp, R_1, \dots, R_7 as well as $\text{lcs}(R_1, \dots, R_7)$, which turned out to be equivalent to an already existing description, we end up with 21 candidates for new concepts. Of these 21 interesting least common subsumers, only 10 have explicitly been computed during the exploration.

During the calls of the “expert”, 255 subsumption tests and 25 n-ary lcs operations have been executed. Because we re-used already computed least common subsumers, the 25 n-ary lcs operations only required 25 binary lcs operations. The number of counterexamples computed by the expert was also 25.

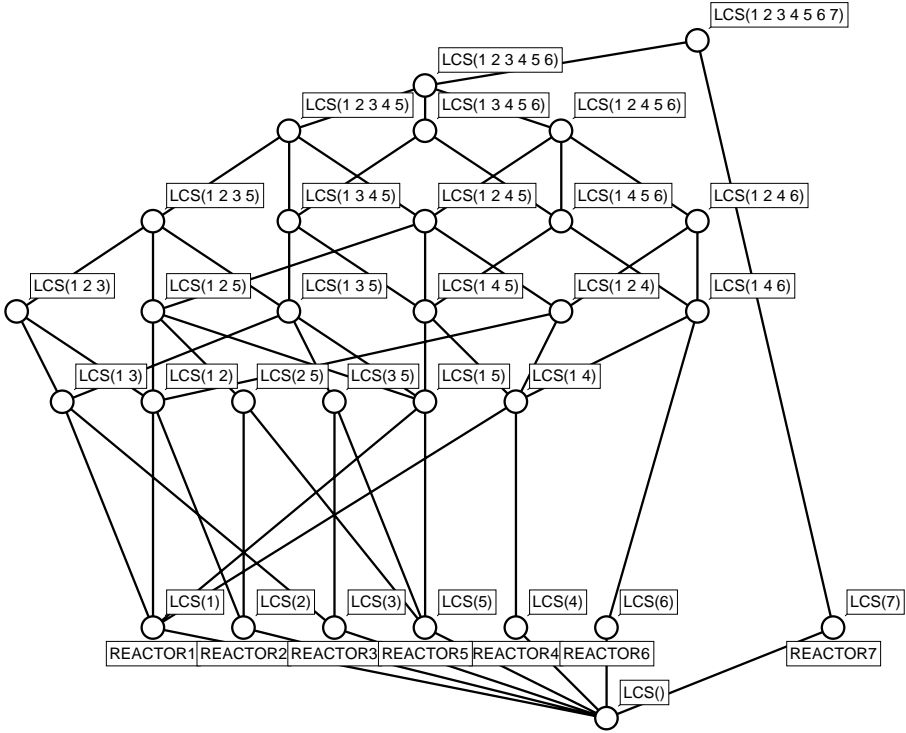


Fig. 1. The hierarchy of least common subsumers of seven reactor descriptions.

Finally, we measured the time needed for executing the interesting subtasks, namely computing the lcs, testing subsumption, and realizing the overhead introduced by the attribute exploration algorithm (e.g., computing the \cdot'' operation, the pseudo-hull, etc). It turned out that more than 84% of the time was used for computing least common subsumers, 15% for subsumption tests, and less than 1% for the rest. This shows that, at least for this small example, the exploration algorithm does not introduce any measurable overhead³. The fact that computing the lcs needed a lot more time than testing subsumption is probably due to the fact that we used a highly optimized subsumption algorithm [22], but only a first prototypical implementation of the lcs algorithm.

What Can Be Learned from the Concept Lattice? Two important facts about the reactor descriptions can be read off immediately. First, there is no subsumption relationship between any of the 7 concepts since all singleton sets occur as intents. Second, Reactor 7 is quite different from the other reactors since its lcs with any

³ This is in strong contrast to the experimental results described in Subsection 6.1. The main difference appears to be that, for the present context, the concept lattice and the number of counterexamples is not much larger than the number of implications in the base.

of the others yields a very general concept description. Thus, it should not be used for generating new concepts together with the other ones. In fact, a closer look at R_7 revealed that, though it describes a reactor of a type similar to that of the other ones, this description was given on a completely different level of abstraction.

Next, let us consider the question of which of the least common subsumers occurring in the lattice appear to be good candidates for providing an interesting new concept. First, the lcs of the whole set is ruled out since it involves Reactor 7, which does not fit well with the other examples (see above). Second, in order to avoid concepts that are too specific, least common subsumers that do not cover more than half of the reactors should also be avoided. If we use these two criteria, then we are left with 9 candidates (the formal concepts with intents of cardinality 4, 5, and 6), which is a number of concepts that can well be inspected by the process engineer. In our example, the 5 least common subsumers on the first layer of these interesting candidates (the formal concepts with intents of cardinality 4) were considered by the process engineers to be the most interesting new concepts among the 9 candidates.

7 A More Abstract Point of View

The results and proofs presented in Subsection 4.2 and in Section 5 are very similar, and the proofs are based on rather generic arguments (i.e., they use almost no specific properties of the lcs or the conjunctions of defined concepts). Thus, one may ask whether the constructions and arguments used there can be generalized. The purpose of this section is to show that this is indeed the case.

Consider a partially ordered set (M, \preceq) for which all finite infima exist, i.e., if B is a finite subset of M , then there exists an element $\inf(B) \in M$ that is the greatest element of M smaller than all elements of B . From the algorithmic point of view we assume that there are algorithms for deciding the relation \preceq and for computing $\inf(B)$ for all finite subsets B of M . In Subsection 4.2, M is the set of all concept descriptions of the DL under consideration, \preceq is subsumption w.r.t. the TBox ($\sqsubseteq_{\mathcal{T}}$), and the infimum of a finite set of such descriptions is their conjunction⁴. In Section 5, M is again the set of all concept descriptions of the DL under consideration, \preceq is inverse subsumption (\sqsupseteq) and the infimum is given by the lcs⁵.

Definition 21 *Given a finite subset N of M , its infimum closure is the set*

$$\text{InfC}(N) := \{\inf(B) \mid B \subseteq N\}.$$

⁴ To be more precise, M is the set of all equivalence classes $[C]_{\equiv}$ of concept descriptions C and the partial order is the partial order \sqsubseteq_{\equiv} induced by subsumption w.r.t. \mathcal{T} on these equivalence classes.

⁵ Since we take inverse subsumption, the lcs, which is the supremum w.r.t. subsumption, is indeed the infimum.

In Subsection 4.2, N is the set of all concept names defined in the TBox, and $\text{InfC}(N)$ is the set of all conjunctions of such names. In Section 5, N is a finite set \mathcal{C} of concept descriptions, and $\text{InfC}(N)$ is the set of all least common subsumers of subsets of \mathcal{C} .

Since N is finite, $(\text{InfC}(N), \preceq)$ is a complete semilattice, and thus a complete lattice. We are interested in computing this lattice. In Subsection 4.2, $(\text{InfC}(N), \preceq)$ is the hierarchy of all conjunctions of defined concepts, and in Section 5, $(\text{InfC}(N), \preceq)$ is the hierarchy of all least common subsumers of subsets of \mathcal{C} . In order to compute $(\text{InfC}(N), \preceq)$, we define a formal context with attribute set N such that the concept lattice of this context is isomorphic to $(\text{InfC}(N), \preceq)$.

Definition 22 *Let (M, \preceq) be a partially ordered set for which all finite infima exist, and let N be a finite subset of M . The formal context $\mathcal{K}_{\preceq}(N) = (\mathcal{O}, \mathcal{P}, \mathcal{S})$ is defined as follows:*

$$\begin{aligned}\mathcal{O} &:= M, \\ \mathcal{P} &:= N, \\ \mathcal{S} &:= \{(m, n) \mid m \preceq n\}.\end{aligned}$$

Valid implications in $\mathcal{K}_{\preceq}(N)$ correspond to \preceq -relationships between infima of subsets of N . The proof of this result is an easy generalization of the proof of Lemma 13.

Lemma 23 *Let B_1, B_2 be subsets of $\mathcal{P} = N$. The implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\preceq}(N)$ iff $\inf(B_1) \preceq \inf(B_2)$.*

Proof. First, we prove the *only if* direction. If the implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\preceq}(N)$, then this means that the following holds for all objects $m \in \mathcal{O}$: if $m \preceq n$ holds for all $n \in B_1$, then $m \preceq n$ also holds for all $n \in B_2$. Thus, if we take $\inf(B_1)$ as object m , we obviously have $\inf(B_1) \preceq n$ for all $n \in B_1$, and thus $\inf(B_1) \preceq n$ for all $n \in B_2$, which shows $\inf(B_1) \preceq \inf(B_2)$.

Second, we prove the *if* direction. If $\inf(B_1) \preceq \inf(B_2)$, then any object m satisfying $m \preceq \inf(B_1)$ also satisfies $m \preceq \inf(B_2)$ by the transitivity of \preceq . Consequently, if $m \preceq n$ for all $n \in B_1$, then $m \preceq \inf(B_1) \preceq \inf(B_2) \preceq n$ for all $n \in B_2$, i.e., if m satisfies all attributes in B_1 , it also satisfies all attributes in B_2 . This shows that the implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\preceq}(N)$. \square

The proof of the following theorem is an easy generalization of the proof of Theorem 10 and Theorem 14.

Theorem 24 *The concept lattice of the context $\mathcal{K}_{\preceq}(N)$ is isomorphic to the lattice $(\text{InfC}(N), \preceq)$.*

Proof. In order to obtain an appropriate isomorphism, we define a mapping π from the formal concepts of the context $\mathcal{K}_{\preceq}(N)$ to $\text{InfC}(N)$ as follows:

$$\pi(A, B) = \inf(B).$$

Since B is a finite subset of $\mathcal{P} = N$, the definition of $\text{InfC}(N)$ implies that $\text{inf}(B) \in \text{InfC}(N)$.

For formal concepts $(A_1, B_1), (A_2, B_2)$ of $\mathcal{K}_{\preceq}(N)$ we have $(A_1, B_1) \leq (A_2, B_2)$ iff $B_2 \subseteq B_1$. Since B_1 is the intent of the formal concept (A_1, B_1) , we have $B_1 = A'_1 = A'''_1 = B''_1$, and thus $B_2 \subseteq B_1$ iff $B_2 \subseteq B''_1$ iff the implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\preceq}(N)$ iff $\text{inf}(B_1) \preceq \text{inf}(B_2)$. Overall, we have thus shown that π is an order embedding (and thus injective): $(A_1, B_1) \leq (A_2, B_2)$ iff $\text{inf}(B_1) \preceq \text{inf}(B_2)$.

It remains to be shown that π is surjective as well. Let B be an arbitrary subset of $\mathcal{P} = N$. We must show that $\text{inf}(B)$ can be obtained as an image under the mapping π . We know that (B', B'') is a formal concept of $\mathcal{K}_{\preceq}(N)$, and thus it is sufficient to show that $\pi(B', B'') = \text{inf}(B)$, i.e., $\text{inf}(B'') = \text{inf}(B)$. Obviously, $B \subseteq B''$ implies $\text{inf}(B'') \preceq \text{inf}(B)$. Conversely, the implication $B \rightarrow B''$ holds in $\mathcal{K}_{\preceq}(N)$, and thus Lemma 23 yields $\text{inf}(B) \preceq \text{inf}(B'')$. Since \preceq is antisymmetric, this shows $\text{inf}(B) = \text{inf}(B'')$. \square

If we want to apply Algorithm 7 to compute the concept lattice and the Duquenne-Guigues base of $\mathcal{K}_{\preceq}(N)$, we need an “expert” for this context. The proof of the next proposition is a generalization of the proofs of Proposition 20 and of Proposition 15.

Proposition 25 *Given a decision procedure for \preceq as well as an algorithm for computing the infima of all finite subsets of M , these algorithms can be used to obtain an expert for the context $\mathcal{K}_{\preceq}(N)$.*

Proof. First, we show how to decide whether a given implication $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\preceq}(N)$ or not. By Lemma 23, we know that $B_1 \rightarrow B_2$ holds in $\mathcal{K}_{\preceq}(N)$ iff $\text{inf}(B_1) \preceq \text{inf}(B_2)$. Obviously, $\text{inf}(B_1) \preceq \text{inf}(B_2)$ iff $\text{inf}(B_1) \preceq n$ for all $n \in B_2$. Thus, to answer the question “ $B_1 \rightarrow B_2$?”, we first compute $\text{inf}(B_1)$ and then use the decision procedure for \preceq to test whether $\text{inf}(B_1) \preceq n$ holds for all $n \in B_2$.

Second, assume that $B_1 \rightarrow B_2$ does not hold in $\mathcal{K}_{\preceq}(N)$, i.e., $\text{inf}(B_1) \not\preceq \text{inf}(B_2)$. We claim that $\text{inf}(B_1)$ is a counterexample, i.e., $\text{inf}(B_1) \in B'_1$ and $\text{inf}(B_1) \notin B'_2$. This is an immediate consequence of the facts that $B'_i = \{m \in \mathcal{O} \mid m \preceq n \text{ for all } n \in B_i\} = \{m \in \mathcal{O} \mid m \preceq \text{inf}(B_i)\}$ ($i = 1, 2$) and that $\text{inf}(B_1) \preceq \text{inf}(B_1)$ and $\text{inf}(B_1) \not\preceq \text{inf}(B_2)$.

Of this counterexample, Algorithm 7 really needs the row corresponding to this object in the matrix corresponding to \mathcal{S} . This row can easily be computed using the decision procedure for \preceq : for each $n \in \mathcal{P} = N$, we use this decision procedure to test whether $\text{inf}(B_1) \preceq n$ holds or not. \square

To sum up, we have shown that attribute exploration can be used to compute (a representation of) the lattice $(\text{InfC}(N), \preceq)$ provided that \preceq is decidable and all finite infima are computable. The results presented in Subsection 4.2 and in Section 5 are instances of this general result. The fact that the approach described in Section 5 (and first presented in [10]) can be generalized in this direction has already been mentioned in [18] (Section 3), but not worked out in detail.

8 Conclusion

We have described two cases where a tool from FCA (attribute exploration) can be used to compute an extended subsumption hierarchy in DL (the hierarchy of conjunctions of concepts defined in a terminology, and the hierarchy of all least common subsumers of a finite set of concept descriptions). The experimental results show that this approach is much better than the naive approach for computing these hierarchies. Nevertheless, there is still room for improvements. For example, in the first case the overhead of the FCA part of the algorithm was quite high due to the large number of formal concepts in the concept lattice. For most applications, one does not really need to compute all formal concepts since the implication base already contains all relevant information. Attribute exploration (as described in Section 3) generates all intents of formal concepts, since it enumerates all pseudo-closed sets, which are either concept intents, left-hand sides of implications in the base, or left-hand sides of implications that are not valid (i.e., implications that yield a counterexamples during the exploration process). In contexts whose concept lattice is quite large compared to the size of the Duquenne-Guigues base and the number of counterexamples, it would be better to have a modified attribute exploration algorithm that enumerates only those pseudo-closed sets that are not concept intents (only for those, the expert is called).

We have also seen that the approaches described in Subsection 4.2 and Section 5 are both instances of a more general approach. Thus, one can try to find other interesting instances of this general approach. One example could be to compute the hierarchy of all conjunctions of defined concepts and their negations. In fact, when considering the lcs in DLs that are more expressive than \mathcal{EL} (e.g., in \mathcal{ALC}), one must deal with such conjunctions. Since in this case one has background knowledge about the relationship between different attributes (the concept A is disjoint from its negation $\neg A$), one can probably employ methods developed for attribute exploration with background knowledge [17].

Acknowledgment

We should like to thank Madjid Nassiri for the implementation of and the experiments with the approach described in Subsection 4.2 and Ralf Molitor for the implementation of and the experiments with the approach described in Section 5.

References

1. Franz Baader. Computing a minimal representation of the subsumption lattice of all conjunctions of concepts defined in a terminology. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, pages 168–178, 1995.
2. Franz Baader. Computing the least common subsumer in the description logic \mathcal{EL} w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 11th International Conference on Conceptual Structures, ICCS 2003*, volume 2746 of *Lecture Notes in Artificial Intelligence*, pages 117–130. Springer-Verlag, 2003.

3. Franz Baader. The instance problem and the most specific concept in the description logic \mathcal{EL} w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 26th Annual German Conference on Artificial Intelligence, KI 2003*, volume 2821 of *Lecture Notes in Artificial Intelligence*, pages 64–78, Hamburg, Germany, 2003. Springer-Verlag.
4. Franz Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 319–324. Morgan Kaufmann, 2003.
5. Franz Baader. Terminological cycles in a description logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 325–330. Morgan Kaufmann, 2003.
6. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
7. Franz Baader, Enrico Franconi, Bernhard Hollunder, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.
8. Franz Baader and Ralf Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic \mathcal{ALN} -concept descriptions. In *Proc. of the 22nd German Annual Conf. on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1998.
9. Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.
10. Franz Baader and Ralf Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In B. Ganter and G. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues – Proceedings of the 8th International Conference on Conceptual Structures (ICCS2000)*, volume 1867 of *Lecture Notes in Artificial Intelligence*, pages 290–303. Springer-Verlag, 2000.
11. Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
12. William W. Cohen and Haym Hirsh. Learning the CLASSIC description logics: Theoretical and experimental results. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 121–133, 1994.
13. William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.
14. Vincent Duquenne. Contextual implications between attributes and some representational properties for finite lattices. In Bernhard Ganter, Rudolf Wille, and Karl Erich Wolf, editors, *Beiträge zur Begriffsanalyse*, pages 213–239. B.I. Wissenschaftsverlag, Mannheim, 1987.
15. Michael Frazier and Leonard Pitt. CLASSIC learning. *Machine Learning*, 25:151–193, 1996.
16. Bernhard Ganter. Finding all closed sets: A general approach. *Order*, 8:283–290, 1991.

17. Bernhard Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999.
18. Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In Harry S. Delugach and Gerd Stumme, editors, *Conceptual Structures: Broadening the Base, Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer-Verlag, 2001.
19. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, 1999.
20. Volker Haarslev and Ralf Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 2001.
21. Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.
22. Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.
23. Ralf Küsters and Alex Borgida. What's in an attribute? Consequences for the least common subsumer. *J. of Artificial Intelligence Research*, 14:167–203, 2001.
24. Ralf Küsters and Ralf Molitor. Approximating most specific concepts in description logics with existential restrictions. In Franz Baader, Gerd Brewka, and Thomas Eiter, editors, *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 33–47, Vienna, Austria, 2001. Springer-Verlag.
25. Ralf Küsters and Ralf Molitor. Computing least common subsumers in \mathcal{ALN} . In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 219–224, 2001.
26. Carsten Lutz. Complexity of terminological reasoning revisited. In *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 181–200. Springer-Verlag, 1999.
27. Wolfgang Marquardt. Trends in computer-aided process modeling. *Computers and Chemical Engineering*, 20(6/7):591–609, 1996.
28. Ralf Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken (Supporting the Modelling of Chemical Processes by Using Non-standard Inferences in Description Logics)*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2000. In German.
29. Madjid Nassiri. Berechnung einer erweiterten Subsumtionshierarchie (Computation of an extended subsumption hierarchy). Diploma thesis, RWTH Aachen, Germany, 1997. In German.
30. Susanne Prediger. Terminologische Merkmalslogik in der Formalen Begriffsanalyse. In [37]. 2000.
31. Susanne Prediger and Gerd Stumme. Theory-driven logical scaling: Conceptual information systems meet description logics. In Enrico Franconi and Michael Kifer, editors, *Proc. of the 6th Int. Workshop on Knowledge Representation meets Databases (KRDB'99)*, 1999.
32. Alan Rector and Ian Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, 1997. AAAI Press.

33. Sebastian Rudolf. An FCA method for the extensional exploration of relational data. In Bernhard Ganter and Aldo de Moor, editors, *Using Conceptual Structures – Contributions to ICCS 2003*. Shaker Verlag, 2003.
34. Ulrike Sattler. *Terminological Knowledge Representation Systems in a Process Engineering Application*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1998.
35. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
36. Stefan Schultz and Udo Hahn. Knowledge engineering by large-scale knowledge reuse—experience from the medical domain. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 601–610. Morgan Kaufmann, 2000.
37. Gerd Stumme and Rudolf Wille. *Begriffliche Wissensverarbeitung – Methoden und Anwendungen*. Springer-Verlag, 2000.
38. Frank Vogt. *Formale Begriffsanalyse mit C++*. Springer-Verlag, 1996.
39. Monika Zickwolff. *Rule Exploration: First Order Logic in Formal Concept Analysis*. PhD thesis, TH Darmstadt, Germany, 1991.

Machine Learning and Formal Concept Analysis

Sergei O. Kuznetsov

All-Russian Institute for Scientific and Technical Information
Technische Universität Dresden

Abstract. A model of learning from positive and negative examples is naturally described in terms of Formal Concept Analysis (FCA). In these terms, result of learning consists of two sets of intents (closed subsets of attributes): the first one contains intents that have only positive examples in the corresponding extents. The second one contains intents such that the corresponding extents contain only negative examples. On the one hand, we show how the means of FCA allows one to realize learning in this model with various data representation, from standard object-attribute one to that with labeled graphs. On the other hand, we use the language of FCA to give natural descriptions of some standard models of Machine Learning such as version spaces and decision trees. This allows one to compare several machine learning approaches, as well as to employ some standard techniques of FCA in the domain of machine learning. Algorithmic issues of learning with concept lattices are discussed. We consider applications of the concept-based learning, including Structure-Activity Relationship problem (in predictive toxicology) and spam filtering.

1 Introduction

Machine Learning is usually defined as a discipline “concerned with the question of how to construct computer programs that automatically improve with experience” [46]. Methods of FCA [62, 23] were from the very beginning (viz., attribute exploration and more later predicate, object and relational explorations) more oriented to human-machine interaction, thus being more along the lines of knowledge discovery: “The KDD process is interactive and iterative, involving numerous steps with many decision being made by the user” [13] and similar principles were e.g., declared in [29, 61]. However, in this paper we would like to relate FCA rather to mathematical models of machine learning, which underlie methods of knowledge discovery. The latter essentially consists in application-driven combination of various learning models under supervision of human experts, which also perform data selection at various stages of the discovery process.

From the very beginning, techniques related to extraction of knowledge from data were among the mainstream of FCA research. Implications between sets of attributes in formal contexts, as opposed to mathematically equivalent functional dependencies in databases, are drawn from datasets, whereas in the DBMS paradigm [41] the database dependencies are usually known in advance to a DB designer. Generating bases of implications from contexts can certainly be called

a machine-learning procedure, the same holds for generation of bases of partial implications [42], which became known later as association rules in data mining.

One of the first models of machine learning that used lattices (closure systems) was the JSM-method¹ of automated hypothesis generation [16, 17]. In this model positive hypotheses are sought among intersections of positive example descriptions (object intents), same for negative hypotheses. Various additional conditions can be imposed on these intersections. For example, in Section 2 we consider so-called counterexample forbidding hypotheses, which are equivalent to implications with the value of the target attribute (positive or negative) in the consequence and closed set of attributes in the premise.

In terms of FCA, the system CHARADE described in [18] basically constructs rules of the form $A \rightarrow A''$. With the use of properties of Galois connections a sort of nonminimal base of implications is obtained. In system GRAND [50] learning with lattices proceeded as follows: At input the system has training positive and negative examples described by many-valued attributes. The system creates partial description and orders them by generality, completes the partial order to a lattice (which is known in the lattice theory as Dedekind-McNeille completion) and then finds implications (in the FCA sense) with consequences being values of the target attributes. In fact a base of implications with minimal premises is obtained, which however is not minimal in the number of implications. Some machine learning systems, e.g., those described in [55, 43] use various heuristics (based on allowances, accuracy, confidence, support, entropy, etc.) for feature selection and reduction of the number of possible concept-based dependencies learned from positive and negative examples. For example, Rulelearner system [55] generates hypotheses (closed sets of attributes that are subsets of only some positive examples) with largest extents, which obviously belong to the set of minimal hypotheses. Among these the system looks for hypotheses with smallest cardinality. Additionally to this, the system deletes useless hypotheses, i.e., those that produce no classification of previously unclassified examples. Then the system deletes those instances that result only in useless hypotheses and repeat hypothesis generation for the new dataset. GALOIS system described in [8] realized clustering (i.e., unsupervised learning) based on concept lattices. For classifying a new object similarity between it and existing clusters (concepts) is computed as the number of common attributes. In 1990s the idea of a version space was elaborated by means of logical programming within the Inductive Logical Programming (ILP), where the notion of a subsumption lattice plays an important role [48]. In late 1990s the notion of a lattice of “closed itemsets” became important in the data mining community, see [13, 52].

The paper is organized as follows. In Section 2 we recall basic definitions of FCA and those related to concept-based hypotheses. In Section 3 we consider a learning model for pattern structures, i.e., for data that cannot be directly described by object-attribute matrices, but allow for a computable “meet” operation. In Section 4 we consider version spaces, a basic construction in machine

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

learning, related to all possible classifiers compatible with a training sample, from the viewpoint of FCA. In Section 5 we show how decision trees can be naturally captured in terms of FCA. In Section 6 we discuss algorithmic issues of concept-based learning. Finally, in Section 7 we consider some applications of concept-based learning.

2 Basic Definitions: Concepts, Implications, and Hypotheses

First, to make the paper self-contained, we introduce standard definitions of Formal Concept Analysis (FCA) [23]. We consider a set M of “structural attributes”, a set G of objects (or observations) and a relation $I \subseteq G \times M$ such that $(g, m) \in I$ if and only if object g has the attribute m . Such a triple $\mathbb{K} := (G, M, I)$ is called a *formal context*. Using the *derivation operators*, defined for $A \subseteq G$, $B \subseteq M$ by

$$\begin{aligned} A' &:= \{m \in M \mid gIm \text{ for all } g \in A\}, \\ B' &:= \{g \in G \mid gIm \text{ for all } m \in B\}, \end{aligned}$$

we can define a *formal concept* (of the context \mathbb{K}) to be a pair (A, B) satisfying $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$. A is called the *extent* and B is called the *intent* of the concept (A, B) . These concepts, ordered by

$$(A_1, B_1) \geq (A_2, B_2) \iff A_1 \supseteq A_2$$

form a complete lattice, called *the concept lattice* of $\mathbb{K} := (G, M, I)$.

Next, we use the FCA setting to describe JSM-hypotheses from [16, 17]. In addition to the structural attributes of M , we consider (as in [36, 19]) a *target attribute* $\omega \notin M$. This partitions the set G of all objects into three subsets: The set G_+ of those objects that are known to have the property ω (these are the *positive examples*), the set G_- of those objects of which it is known that they do not have ω (the *negative examples*) and the set G_τ of *undetermined examples*, i.e., of those objects, of which it is unknown if they have property ω or not. This gives three subcontexts of $\mathbb{K} = (G, M, I)$, the first two staying for the training sample:

$$\mathbb{K}_+ := (G_+, M, I_+), \quad \mathbb{K}_- := (G_-, M, I_-), \quad \text{and } \mathbb{K}_\tau := (G_\tau, M, I_\tau),$$

where for $\varepsilon \in \{+, -, \tau\}$ we have $I_\varepsilon := I \cap (G_\varepsilon \times M)$ and the corresponding derivation operators are denoted by $(\cdot)^+$, $(\cdot)^-$, $(\cdot)^\tau$, respectively.

Intents, as defined above, are attribute sets shared by some of the observed objects. In order to form hypotheses about structural causes of the target attribute ω , we are interested in sets of structural attributes that are common to some positive, but to no negative examples. Thus, a *positive hypothesis* h for ω (called “counter-example forbidding hypotheses” in the JSM-method [16, 17]) is an intent of \mathbb{K}_+ such that $h^+ \neq \emptyset$ and $h \not\subseteq g^- := \{m \in M \mid (g, m) \in I_-\}$ for any negative example $g \in G_-$. An intent of \mathbb{K}_+ that is contained in the intent of a

negative example is called a *falsified (+)-generalization*. *Negative hypotheses* are defined similarly. 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 standard probabilistic techniques known in machine learning and data mining (majority vote, Bayesian approach, etc.)

In [35, 36] 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.

Example 1. Consider the following data table

G \ M	color	form	rigid	smooth	target
1 apple	yellow	round	no	yes	+
2 grapefruit	yellow	round	no	no	+
3 kiwi	green	oval	no	no	+
4 plum	blue	oval	no	yes	+
5 toy cube	green	cubic	yes	yes	—
6 egg	white	oval	yes	yes	—
7 tennis ball	white	round	no	no	—

This dataset or *multivalued context* can be reduced to a context of the form presented above by *scaling* [23], e.g., as follows (scaling 1):

G \ M	w	y	g	b	f	\bar{f}	s	\bar{s}	r	o	\bar{o}	target
1 apple		×			×	×	×	×	×	×		+
2 grapefruit		×			×		×	×	×	×		+
3 kiwi			×		×		×		×			+
4 plum				×	×	×			×			+
5 toy cube			×		×		×			×		—
6 egg	×				×		×			×		—
7 tennis ball	×				×		×	×				—

Here we use the following abbreviations: “w” for white, “y” for yellow, “g” for green, “b” for blue, “s” for smooth, “f” for firm, “r” for round, “o” for oval, and “ \bar{m} ” for $m \in \{w, y, g, b, s, f, r, o\}$. This context gives rise to the positive concept lattice in Fig. 1, where we marked minimal (+)-hypotheses and falsified (+)-generalizations. If we have an undetermined example **mango** with $\text{mango}^\tau = \{y, \bar{f}, s, o\}$ then it is classified positively, since mango^τ contains the minimal hypothesis $\{\bar{f}, o\}$ and does not contain any negative hypothesis.

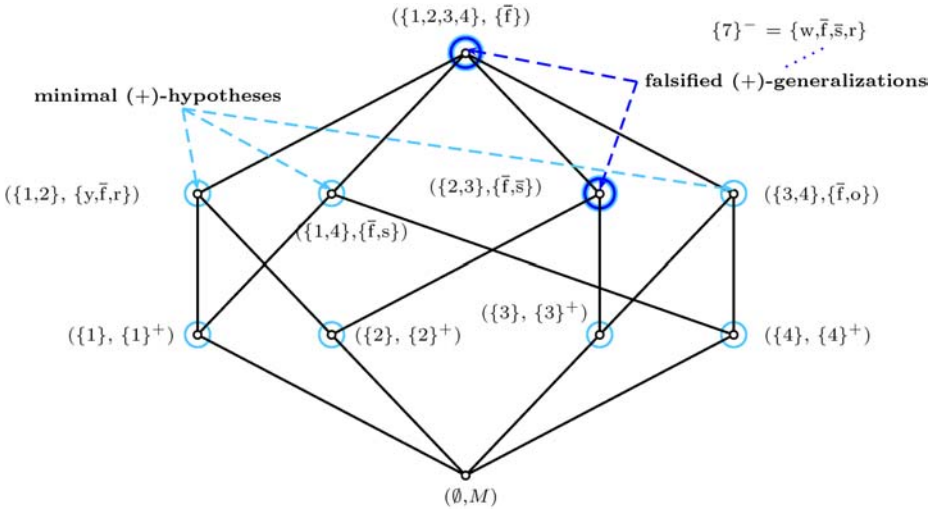


Fig. 1. Positive concept lattice for scaling 1

For this scaling we have two minimal negative hypotheses: $\{w\}$ (supported by examples **egg** and **tennis ball** and $\{f, s\}$ (supported by examples **toy cube** and **egg**. The context can be scaled differently, e.g. in this way (scaling 2):

$G \setminus M$	w	y	g	b	\overline{w}	\overline{y}	\overline{g}	\overline{b}	f	\overline{f}	s	\overline{s}	r	o	\overline{r}	\overline{o}	target
1 apple	×		×	×	×	×	×	×	×	×	×	×	×	×	×	×	+
2 grapefruit	×		×	×	×	×	×	×	×	×	×	×	×	×	×	×	+
3 kiwi		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	+
4 plum			×	×	×	×	×	×	×	×	×	×	×	×	×	×	+
5 toy cube		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	−
6 egg	×				×	×	×	×	×	×	×	×	×	×	×	×	−
7 tennis ball	×				×	×	×	×	×	×	×	×	×	×	×	×	−

This scaling gives rise to another positive concept lattice, all intents of which are (+)-hypotheses. The unique minimal hypothesis (corresponding to the top element of the concept lattice) is $\{\overline{w}, \overline{f}, o\}$. Two minimal negative hypotheses are $\{\overline{y}, \overline{b}, \overline{r}, f, s\}$ (supported by examples 5 and 6) and $\{\overline{y}, \overline{g}, \overline{b}, w, o\}$ (supported by examples 6 and 7).

3 Learning in Pattern Structures

3.1 Pattern Structures and Hypotheses therein

Learning with descriptions given by logical formulas is systematically studied in ILP [48], with applications to learning with molecular graphs described by logical formulas [60, 6]. In FCA community several authors have considered the

case where instead of having attributes the objects satisfy certain logical formulas [2, 9, 14] or they are described by labeled graphs [35, 37, 40]. In case of logical formulas shared attributes are replaced by common subsumers of the respective formulas. In [20] we showed how such an approach is linked to the general FCA framework.

Let G be some set, let (D, \sqcap) be a meet-semilattice and let $\delta : G \rightarrow D$ be a mapping. Then $(G, \underline{D}, \delta)$ with $\underline{D} = (D, \sqcap)$ is called a *pattern structure*, provided that the set

$$\delta(G) := \{\delta(g) \mid g \in G\}$$

generates a complete subsemilattice (D_δ, \sqcap) of (D, \sqcap) , i.e., every subset X of $\delta(G)$ has an infimum $\sqcap X$ in (D, \sqcap) and D_δ is the set of these infima. Each such complete semilattice has lower and upper bounds, which we denote by $\mathbf{0}$ and $\mathbf{1}$, respectively. There are two natural situations where the condition on the complete subsemilattice is automatically satisfied: when (D, \sqcap) is complete, and when G is finite.

If $(G, \underline{D}, \delta)$ is a pattern structure, we define the derivation operators as

$$A^\diamond := \sqcap_{g \in A} \delta(g) \quad \text{for } A \subseteq G$$

and

$$d^\diamond := \{g \in G \mid d \sqsubseteq \delta(g)\} \quad \text{for } d \in D.$$

The elements of D are called *patterns*. The natural order on them is given, as usual, by

$$c \sqsubseteq d : \Longleftrightarrow c \sqcap d = c,$$

and is called the *subsumption* order². The operators \diamond obviously make a Galois connection between the power set of G and (D, \sqsubseteq) . The pairs (A, d) satisfying

$$A \subseteq G, \quad d \in D, \quad A^\diamond = d, \quad \text{and} \quad A = d^\diamond$$

are called the *pattern concepts* of $(G, \underline{D}, \delta)$, with extent A and *pattern intent* d . For $a, b \in D$ the *pattern implication* $a \rightarrow b$ holds if $a^\diamond \sqsubseteq b^\diamond$. Similarly, for $C, D \subseteq G$ the *object implication* $C \rightarrow D$ holds if $C^\diamond \sqsubseteq D^\diamond$.

Since (D_δ, \sqcap) is complete, there is a (unique) operation \sqcup such that $(D_\delta, \sqcap, \sqcup)$ is a complete lattice. It is given by

$$\sqcup X := \sqcap \{c \in D_\delta \mid \forall_{x \in X} x \sqsubseteq c\}.$$

A subset M of D is \sqcup -dense for (D_δ, \sqcap) if every element of D_δ is of the form $\sqcup X$ for some $X \subseteq M$. If this is the case, then with

$$\downarrow d := \{e \in D \mid e \sqsubseteq d\}$$

we get

² Note that subsumption order on patterns defined in this way (a larger pattern subsumes a smaller pattern), being “intentional,” is inverse to definitions of subsumption in logics, where it is “extensional” (a more general formula, covering more ground facts, subsumes a less general formula).

$$c = \sqcup(\downarrow c \cap M) \quad \text{for every } c \in D_\delta.$$

Of course, $M := D_\delta$ is always an example of a \sqcup -dense set.

If M is \sqcup -dense in (D_δ, \sqcap) , then the formal context (G, M, I) with I given as $gIm: \Leftrightarrow m \sqsubseteq \delta(g)$ is called a *representation context* for $(G, \underline{D}, \delta)$. The following result from [20] can be proved by a standard application of the basic theorem of FCA [23].

Theorem 1 *Let $(G, \underline{D}, \delta)$ be a pattern structure and let (G, M, I) be a representation context of $(G, \underline{D}, \delta)$. Then for any $A \subseteq G$, $B \subseteq M$ and $d \in D$ the following two conditions are equivalent*

1. (A, d) is a pattern concept of $(G, \underline{D}, \delta)$ and $B = \downarrow d \cap M$.
2. (A, B) is a formal concept of (G, M, I) and $d = \sqcup B$.

Thus, the pattern concepts of $(G, \underline{D}, \delta)$ are in 1-1-correspondence with the formal concepts of (G, M, I) . Corresponding concepts have the same first components (called *extents*). These extents form a closure system on G and thus a complete lattice, which is isomorphic to the concept lattice of (G, M, I) .

An approach to generating pattern concepts and implications between objects can be made in lines of a procedure proposed in [2]. This procedure, called the *object exploration*, is the dual of the *attribute exploration* algorithm, which is standard in Formal Concept Analysis [23]. In the beginning of the exploration process one has the empty set of object implications and the set of extents E , consisting at the initialization step of the empty extent. One considers the set of implications of the form $A \rightarrow A''$ for $A \in E$ in the lexicographical order and asks an expert whether each particular implication holds. If the expert says yes, then either the set of implications or the set of extents are updated (dependent on the fact whether a set of objects is pseudoclosed or closed), if not, the expert should provide a counterexample that updates the current set of objects.

As the result of object exploration one obtains the context with the same concept lattice as the lattice of the subsumption hierarchy (in case of description languages this is given by the lattice of least common subsumers) and the stem base of object implications. The procedure proposed in [2] also applies to the general setting with an arbitrary semilattice D .

In [36, 37, 19] we considered a learning model from [17] in terms of Formal Concept Analysis. This model assumes that the cause of a *target property* resides in common attributes of objects that have this property.

For pattern structures this can be formalized as follows. Let $(G, \underline{D}, \delta)$ be a pattern structure together with an external target property ω . As in the case of standard contexts, the set G of all objects is partitioned into three disjoint sets w.r.t. ω : the sets G_+ , G_- , G_τ of positive, negative, and undetermined examples. This gives three pattern substructures of $(G, \underline{D}, \delta)$: $(G_+, \underline{D}, \delta)$, $(G_-, \underline{D}, \delta)$, $(G_\tau, \underline{D}, \delta)$.

A *positive hypothesis* h is defined as a pattern intent of $(G_+, \underline{D}, \delta)$ that is not subsumed by any pattern from $\delta(G_-)$ (for short: not subsumed by any negative example). Formally: $h \in D$ is a positive hypothesis iff

$$h^\diamond \cap G_- = \emptyset \text{ and } \exists A \subseteq G_+ : A^\diamond = h.$$

A *negative hypothesis* is defined accordingly.

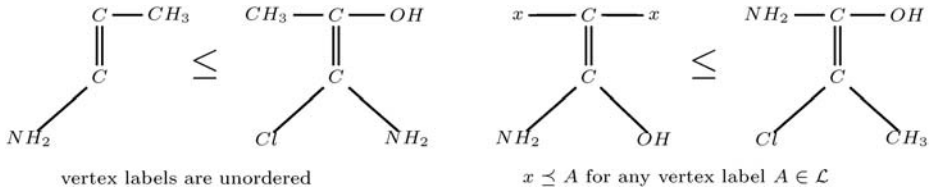
A hypothesis in the sense of Section 2 [19] is obtained as a special case of this definition when $(D, \sqcap) = (2^M, \cap)$ for some set M . Hypotheses can be used for classification of undetermined examples as introduced in [17] in the following way. If $g \in G_\tau$ is an undetermined example, then a hypothesis h with $h \sqsubseteq \delta(g)$ is *for the positive classification* of g if h is positive and *for the negative classification* of g if it is a negative hypothesis. Classification of an example $g \in G_\tau$ is defined in the same way as in Section 2 (with \subseteq replaced by \sqsubseteq).

Example 2. Consider a pattern structure based on the following ordered set P of graphs with labels from the set \mathcal{L} with partial order \preceq . Each labeled graph Γ from P is a triple of the form $((V, l), E)$, where V is a set of vertices, E is a set of edges and $l: V \rightarrow \mathcal{L}$ is a label assignment function, taking a vertex to its label.

For two graphs $\Gamma_1 := ((V_1, l_1), E_1)$ and $\Gamma_2 := ((V_2, l_2), E_2)$ from P Γ_1 **dominates** Γ_2 or $\Gamma_2 \leq \Gamma_1$ if there exists a one-to-one mapping $\varphi: V_2 \rightarrow V_1$ such that it

- respects edges: $(v, w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$,
- fits under labels: $l_2(v) \preceq l_1(\varphi(v))$.

For example, if $\mathcal{L} = \{C, NH_2, CH_3, OH, x\}$ we can have



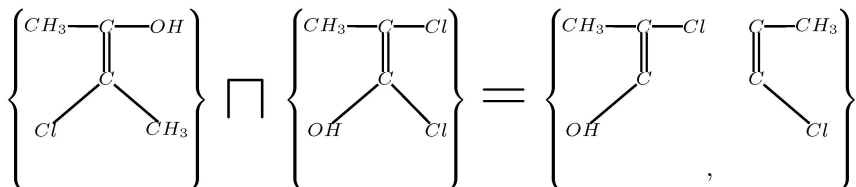
A meet operation \sqcap on graph sets can then be defined as follows: For two graphs X and Y from P

$$\{X\} \sqcap \{Y\} := \{Z \mid Z \leq X, Y, \quad \forall Z_* \leq X, Y \quad Z_* \not\leq Z\},$$

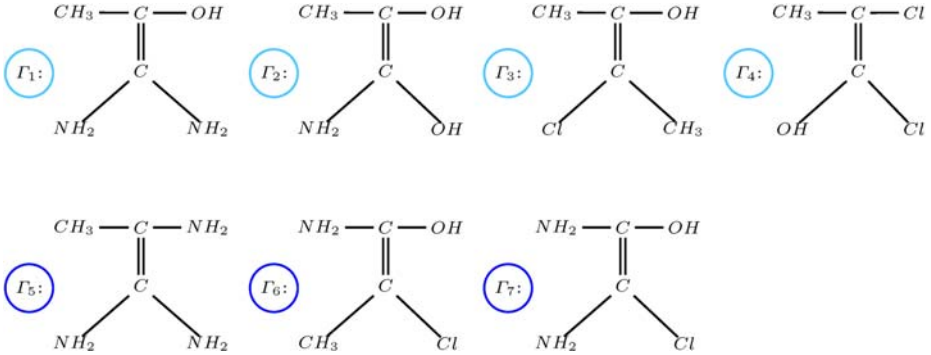
i.e., $\{X\} \sqcap \{Y\}$ is the set of all maximal common subgraphs of X and Y up to substitution of a vertex label by a vertex label smaller w.r.t. \preceq . The meet of nonsingleton sets of graphs is defined as

$$\{X_1, \dots, X_k\} \sqcap \{Y_1, \dots, Y_m\} := \text{MAX}_{\leq}(\sqcup_{i,j} (\{X_i\} \sqcap \{Y_j\}))$$

for details see [35, 37, 20]. Here is an example of applying \sqcap defined above:



Let positive examples be described by graphs $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ and negative examples be described by graphs $\Gamma_5, \Gamma_6, \Gamma_7$:



then the lattice of the pattern structure $(G_+, \underline{D}, \delta)$, where \underline{D} is the semilattice on graph sets and δ is a function taking an object to its graph description, is given in Fig. 2, where (+)-hypotheses and falsified (+)-generalizations are marked:

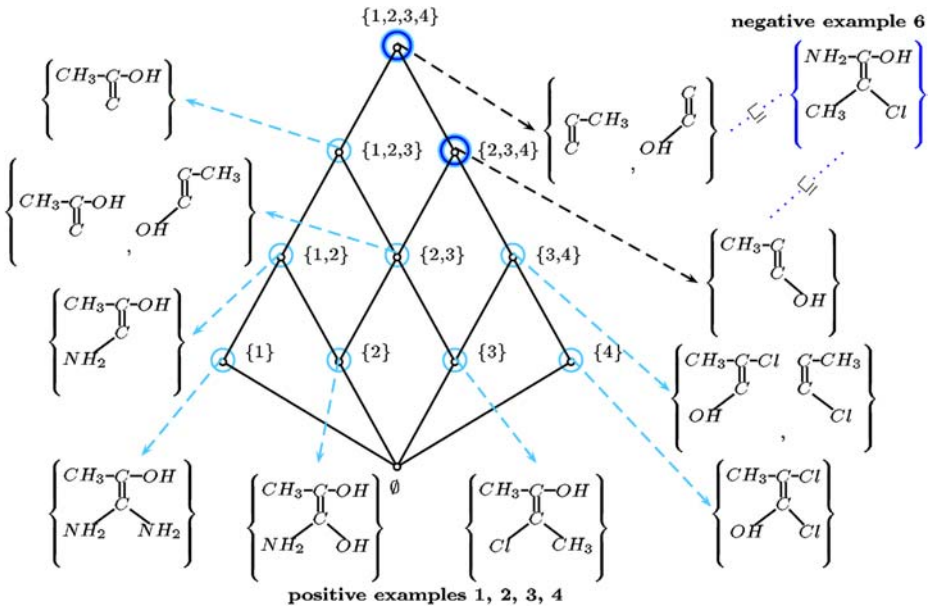


Fig. 2. The lattice of the positive pattern structure

3.2 Projections and Projected Hypotheses

Since for some pattern structures (e.g., for the pattern structure given by sets of graphs with labeled vertices) even computing subsumption relation may be NP-hard, in practical situations we need to look for some approximation tools, which

would replace the patterns with simpler ones, even if that results in some loss of information. To this end we use a mapping $\psi: D \rightarrow D$ that replaces each pattern $d \in D$ by $\psi(d)$ such that the pattern structure $(G, \underline{D}, \delta)$ is replaced by $(G, \underline{D}, \psi \circ \delta)$. To distinguish two pattern structures, which we consider simultaneously, we use the symbol \diamond only for $(G, \underline{D}, \delta)$, not for $(G, \underline{D}, \psi \circ \delta)$. We additionally require that ψ is a kernel operator (or **projection**), i.e., that ψ is

monotone: if $x \sqsubseteq y$, then $\psi(x) \sqsubseteq \psi(y)$,

contractive: $\psi(x) \sqsubseteq x$, and

idempotent: $\psi(\psi(x)) = \psi(x)$.

This requirement seems to hold for any natural approximation mapping and projection thus defined has a nice property well-known in order theory: Any projection of a complete semilattice (D, \sqcap) is \sqcap -preserving, i.e., for any $X, Y \in D$

$$\psi(X \sqcap Y) = \psi(X) \sqcap \psi(Y).$$

This helps us to describe how the lattice of pattern concepts changes when we replace $(G, \underline{D}, \delta)$ by its approximation $(G, \underline{D}, \psi \circ \delta)$. First, we note that $\psi(d) \sqsubseteq \delta(g) \Leftrightarrow \psi(d) \sqsubseteq \psi \circ \delta(g)$. Then, using the basic theorem of FCA (which, in particular allows one to represent every lattice as a concept lattice), we showed how the projected pattern lattice is represented by a context [20]:

Theorem 2 *For pattern structures $(G, \underline{D}, \delta_1)$ and $(G, \underline{D}, \delta_2)$ the following statements are equivalent:*

1. $\delta_2 = \psi \circ \delta_1$ for some projection ψ of \underline{D} .
2. There is a representation context (G, M, I) of $(G, \underline{D}, \delta_1)$ and some $N \subseteq M$ such that $(G, N, I \cap (G \times N))$ is a representation context of $(G, \underline{D}, \delta_2)$.

Again, the basic theorem helped us to “binarize” the initial data representation. However, to do this, we need first to compute the pattern lattice. Pattern structures are naturally ordered by projections: $(G, \underline{D}, \delta_1) \geq (G, \underline{D}, \delta_2)$ if there is a projection ψ such that $\delta_2 = \psi \circ \delta_1$. In this case, representation $(G, \underline{D}, \delta_2)$ can be said to be rougher than $(G, \underline{D}, \delta_1)$ and the latter to be finer than the former. In comparable pattern structures implications are related as follows: If $\psi(a) \rightarrow \psi(b)$ and $\psi(b) = b$ then $a \rightarrow b$ for arbitrary $a, b \in D$.

The properties of projection allow one to relate hypotheses in the original representation with those approximated by a projection. As in [20] we use the term “hypothesis” to those obtained for $(G, \underline{D}, \delta)$ and we refer to those obtained for $(G, \underline{D}, \psi \circ \delta)$ as ψ -hypotheses. There is no guarantee that the ψ -image of a hypothesis must be a ψ -hypothesis. In fact, our definition allows that ψ is the “null projection” with $\psi(d) = \mathbf{0}$ for all $d \in D$. (total abandoning of the data with no interesting hypotheses). However, if $\psi(d)$ is a (positive) hypothesis, then $\psi(d)$ is also a (positive) ψ -hypothesis. If we want to look another way round, we have the following: if $\psi(d)$ is a (positive) ψ -hypothesis, then $\psi(d)^\diamond$ is a (positive) hypothesis [20].

The set of all hypothesis-based classifications does not shrink when we pass from d to $\psi(d)$. Formally, if d is a hypothesis for the positive classification of g

and $\psi(d)$ is a positive ψ -hypothesis, then $\psi(d)$ is for the positive classification of g .

The above observations show that we can generate hypotheses starting from projections. For example, we can select only those that can be seen in the projected data, which is suggested by the following theorem from [20]:

Theorem 3 *For any projection ψ and any positive hypothesis $d \in D$ the following are equivalent:*

1. $\psi(d)$ is not subsumed by any negative example.
2. There is some positive ψ -hypothesis h such that $h^{\circ\circ} \sqsubseteq d$.

3.3 Algorithmic Problems of Learning in Concept Lattices and Pattern Structures

Computing concept-based hypotheses can be hard. The number of concepts of a formal context, i.e., the size of a concept lattice, can be exponential in the size of a context (e.g., for the context (A, A, \neq) , which gives rise to a Boolean concept lattice) and the problem of computing the size of a concept lattice is $\#P$ -complete [34, 38]. All hypotheses can be generated by a polynomial delay algorithm, however a (cumulative) polynomial delay algorithm for minimal hypotheses is not known. In certain cases, e.g., when the number of attributes per object is bounded, the computation of hypotheses can be realized in polynomial time. In principle, any known algorithm for computing concepts (see, e.g., review [39]) can be adapted to computing hypotheses.

When a pattern structure is given, we may, for example, determine its concepts by computing all infima of subsets of D_δ and thereby all pattern concepts. To this end we can adapt some standard FCA algorithms, e.g., **Next Closure** [23]. Here one should take into account that performing a single closure may take exponential time. For example, already the problem of testing the \sqsubseteq relation for a lattice on sets of labeled graphs from [35, 37, 20] is NP-complete (equivalent to SUBGRAPH ISOMORPHISM problem [24]), and computing $X \sqcap Y$ is even more difficult. A similar algorithm of this type was described in [37] for computing with sets of graphs. The time complexity of the algorithm is $O((\alpha + \beta|G|)|G||L|)$ and its space complexity is $O(\gamma|G||L|)$, where α is time needed to perform \sqcap operation and β is time needed to test \sqsubseteq relation and γ is the space needed to store the largest object from D_δ . Computing the line diagram of the set of all concepts, given the tree generated by the previous algorithm, takes $O((\alpha|G| + \beta|G|^2)|L|)$ time and $O((\gamma|G||L|)$ space [37].

4 Hypotheses, Concept Lattices, and Decision Trees

In this Section we consider the relation between decision trees, concept lattices and concept-based hypotheses. We describe a typical procedure of constructing a decision tree (see, e.g., [54]) in terms of concept lattices.

As input, a system constructing a decision tree receives descriptions of positive and negative examples (or positive and negative contexts). The root of the tree corresponds to the beginning of the process and is not labeled. Other vertices of the decision tree are labeled by attributes and edges are labeled by values of the attributes (e.g., 0 or 1 in case of binary contexts), each leaf is additionally labeled by a class $+$ or $-$, meaning that all examples with attribute values from the path leading from the root to the leaf belong to a certain class, either $+$ or $-$.

Systems like ID3 [54] (see also [46]) compute the value of the *information gain* (or negentropy) for each vertex and each attribute not chosen in the branch above. The attribute with the greatest value of the information gain (with the smallest entropy, respectively) “most strongly separates” objects from classes $+$ and $-$. The algorithm sequentially extends branches of the tree by choosing attributes with the highest information gain. The extension of a branch stops when a next attribute value together with attributes above in the branch uniquely classify examples with this value combination in one of classes $+$ or $-$. In some algorithms, the process of extending a branch stops before this in order to avoid *overfitting*, i.e., the situation where all or almost all examples from the training sample are classified correctly by the resulting decision tree, but objects from test datasets are classified with many errors.

Now we consider decision trees more formally. Let the training data be described by the context $\mathbb{K}_{+-} = (G_+ \cup G_-, M, I_+ \cup I_-)$ with the derivation operator denoted by $(\cdot)'$. In FCA terms this context is called the *subpositional* of \mathbb{K}_+ and \mathbb{K}_- . Assume for simplicity sake that for each attribute $m \in M$ there is an attribute $\overline{m} \in M$, a “negation” of m : $\overline{m} \in g'$ iff $m \notin g'$. A set of attributes M with this property is called *dichotomized* in FCA. We call a subset of attributes $A \subseteq M$ *noncontradictory* if either $m \notin A$ or $\overline{m} \notin A$. We call a subset of attributes $A \subseteq M$ *complete* if for every $m \in M$ one has $m \in A$ or $\overline{m} \in A$.

We would like first to avoid mentioning the use of any optimization functional like information gain for selecting attributes and consider construction of all possible decision trees. The construction of an arbitrary decision tree proceeds by sequentially choosing attributes. If different attributes m_1, \dots, m_k were chosen one after another, then the sequence $\langle m_1, \dots, m_k \rangle$ is called a *decision path* if $\{m_1, \dots, m_k\}$ is noncontradictory and there exists an object $g \in G_+ \cup G_-$ such that $\{m_1, \dots, m_k\}' \subseteq g'$ (i.e., there is an example with this set of attributes). A decision path $\langle m_1, \dots, m_i \rangle$ is a (proper) subpath of a decision path $\langle m_1, \dots, m_k \rangle$ if $i \leq k$ ($i < k$, respectively). A decision path $\langle m_1, \dots, m_k \rangle$ is called *full* if all objects having attributes $\{m_1, \dots, m_k\}$ are either positive or negative examples (i.e., have either $+$ or $-$ value of the target attribute). We call a full decision path *irredundant* if none of its subpaths is a full decision path. The set of all chosen attributes in a full decision path can be considered as a sufficient condition for an object to belong to a class $\varepsilon \in \{+, -\}$. A decision tree is then a set of full decision paths.

In what follows, we shall use extensively the one-to-one correspondence between vertices of a decision tree and the related decision paths, representing the

latter, when this does not lead to ambiguity, by their last chosen attributes. By *closure of a decision path* $\langle m_1, \dots, m_k \rangle$ we mean the closure of the corresponding set of attributes, i.e., $\{m_1, \dots, m_k\}''$. Now we relate decision trees with the covering relation graph of the concept lattice of the context $\mathbb{K} = (G, M, I)$, where the set of objects G is of size $2^{|M|/2}$ and the relation I is such that the set of object intents is exactly the set of complete noncontradictory subsets of attributes. In terms of FCA [23] the context \mathbb{K} is the *semiproduct* of $|M|/2$ *dichotomic scales* or $\mathbb{K} = D_1 \bar{\times} \dots \bar{\times} D_{|M|/2}$ (denoted by $\bar{\times}_M D$ for short), where each dichotomic scale D_i stays for the pair of attributes (m, \bar{m}) .

In a concept lattice a sequence of concepts with decreasing extents we call a *descending chain*. If the chain starts at the top element of the lattice, we call it *rooted*.

Proposition 4. *Every decision path is a rooted descending chain in $\mathfrak{B}(\bar{\times}_M D)$ and every rooted descending chain consisting of concepts with nonempty extents in $\mathfrak{B}(\bar{\times}_M D)$ is a decision path.*

To relate decision trees to hypotheses introduced above we consider again the contexts $\mathbb{K}_+ = (G_+, M, I_+)$, $\mathbb{K}_- = (G_-, M, I_-)$, and $\mathbb{K}_{+-} = (G_+ \cup G_-, M, I_+ \cup I_-)$. The context \mathbb{K}_{+-} can be much smaller than $\bar{\times}_M D$ because the latter always has $2^{|M|/2}$ objects while the number of objects in the former is the number of examples. Also the lattice $\mathfrak{B}(\mathbb{K}_{+-})$ can be much smaller than $\mathfrak{B}(\bar{\times}_M D)$.

Proposition 5. *A full decision path $\langle m_1, \dots, m_k \rangle$ corresponds to a rooted descending chain $\langle (m_1'', m_1'), \dots, (\{m_1, \dots, m_k\}'', \{m_1, \dots, m_k\}') \rangle$ of the line diagram of $\mathfrak{B}(\mathbb{K}_{+-})$ and the closure of each full decision path $\langle m_1, \dots, m_k \rangle$ is a hypothesis, either positive or negative. Moreover, for each minimal hypothesis h , there is a full irredundant path $\langle m_1, \dots, m_k \rangle$ such that $\{m_1, \dots, m_k\}'' = h$.*

This proposition also illustrates the difference between hypotheses and irredundant decision trees. The former correspond to “most cautious” (most specific) learning in the sense that they are least general generalizations of descriptions of positive examples (or object intents, in terms of FCA). The shortest decision paths (for which in no decision tree there exist full paths with proper subsets of attribute values) correspond to the “most courageous” learning (often referred to as “most discriminant” in machine learning community): being the shortest possible rules, they are most general generalizations of positive example descriptions. However, it is not guaranteed that for a given training set resulting in a certain set of minimal hypothesis there is a decision tree such that minimal hypotheses are among closures of its paths (see Example 3 below). In general, to obtain all minimal hypotheses as closures of decision paths one needs to consider several decision trees, not all of them being optimal w.r.t. a procedure based on the information gain functional (like ID3 or C4.5). The issues of generality of generalizations and, in particular, the relation between most cautious (most specific) and most courageous (most general) generalizations, are naturally captured in terms of version spaces, which we consider in the next section.

In real systems for the construction of decision trees like ID3 or C4.5 the process of constructing a decision path is driven by the information gain functional: a

next chosen attribute should have maximal information gain. For dichotomized attributes the information gain is defined for a pair of attributes $m, \overline{m} \in M$. Given a decision path $\langle m_1, \dots, m_k \rangle$

$$\text{IG}(m) := -\frac{|A'_m|}{|G|} \text{Ent}(A_m) - \frac{|A'_{\overline{m}}|}{|G|} \text{Ent}(A_{\overline{m}}),$$

where $A_m := \{m_1, \dots, m_k, m\}$, $A_{\overline{m}} := \{m_1, \dots, m_k, \overline{m}\}$, and for $A \subseteq M$

$$\text{Ent}(A) := - \sum_{\varepsilon \in \{+, -\}} p(\varepsilon \mid A) \cdot \log_2 p(\varepsilon \mid A),$$

$\{+, -\}$ are values of the target attribute and $p(\varepsilon \mid A)$ is the conditional sample probability (for the training set) that an object having a set of attributes A belongs to a class $\varepsilon \in \{+, -\}$. If the derivation operator $(\cdot)'$ is associated with the context $(G_+ \cup G_-, M, I_+ \cup I_-)$, then, by definition of the conditional probability, we have

$$p(\varepsilon \mid A) = \frac{|A' \cap G_\varepsilon|}{|A'|} = \frac{|(A'')' \cap G_\varepsilon|}{|(A'')'|} = p(\varepsilon \mid A'')$$

by the property of the derivation operator $(\cdot)'$: $(A'')' = A'$. This observation implies that instead of considering decision paths, one can consider their closures without affecting the values of the information gain. In terms of lattices this means that instead of the concept lattice $\mathfrak{B}(\mathbb{X}_M D)$ one can consider the concept lattice of the context $\mathbb{K}_{+-} = (G_+ \cup G_-, M, I_+ \cup I_-)$. Another consequence of the invariance of IG w.r.t. closure is the following fact: If implication $m \rightarrow n$ holds in the context $\mathbb{K}_{+-} = (G_+ \cup G_-, M, I_+ \cup I_-)$, then an IG-based algorithm will not choose attribute n in the branch below chosen m and will not choose m in the branch below chosen \overline{n} .

Example 3. Consider the training set from Example 1. The decision tree obtained by the IG-based algorithm is given in Fig. 3. Note that attributes f and w has the same IG value (a similar tree with f at the root is also optimal), the IG-based algorithms usually take the first attribute with the same value of IG.

The decision tree in Fig. 3 corresponds to three implications $\{w\} \rightarrow -$, $\{\overline{w}, f\} \rightarrow -$, $\{\overline{w}, \overline{f}\} \rightarrow +$, such that closures of their premises make the corresponding negative and positive hypotheses for the second scaling from Example 1. Note that the hypothesis $\{\overline{w}, f\}''$ is not minimal, since there is a minimal hypothesis $\{f\}''$ contained in it. The minimal hypothesis $\{f\}''$ corresponds to a decision path of the mentioned IG-based tree with the attribute f at the root.

5 Version Spaces vs. Concept-Based Hypotheses

5.1 Version Spaces

The term “version space” was coined by T. Mitchell [44–46] to denote a variety of models compatible with the training sample of positive and negative examples. Version spaces can be defined in different ways: e.g., in terms of sets of

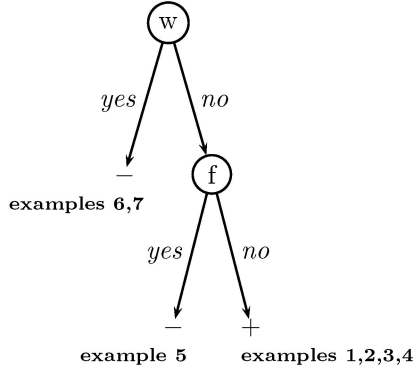


Fig. 3. A decision tree for the dataset from Example 1

maximal and minimal elements [44, 45] or in terms of minimal elements and sets of negative examples [59]. They can also be defined in terms of some matching predicate. These representations are equivalent, however transformations from one into another are not always polynomially tractable. We will start from the representation with matching predicates, in terms slightly modified as compared with [44, 59], in order to avoid collision of FCA terminology and that of machine learning.

- An *example language* L_e (elsewhere also called *instance language*) by means of which the examples (instances) are described. This language describes a set E of *examples*.
- A *classifier language* L_c describing the possible classifiers (elsewhere called *concepts*). This language describes a set C of *classifiers*.
- A *matching predicate* $M(c, e)$ that defines if a classifier c does or does not *match* an example e : We have $M(c, e)$ iff e is an example of classifier c . The set of classifiers is (partially) ordered by a *subsumption order*: for $c_1, c_2 \in L_c$ the classifier c_1 subsumes c_2 or $c_1 \sqsupseteq c_2$ if c_1 corresponds to a more specific description and thus, covers less objects than c_2 :

$$c_1 \sqsupseteq c_2 : \Longleftrightarrow \forall e \in E \ M(c_1, e) \rightarrow M(c_2, e).$$

The corresponding strict order \sqsubset is called *proper subsumption*.

- Sets E_+ and E_- of *positive* and *negative examples* of a *target attribute* with $E_+ \cap E_- = \emptyset$. The target attribute is not explicitly given.
- *consistency predicate* $\text{cons}(c)$:
 $\text{cons}(c)$ holds if for every $e \in E_+$ the matching predicate $M(c, e)$ holds and for every $e \in E_-$ the negation $\neg M(c, e)$ holds. The set of all consistent classifiers is called the *version space*

$$\text{VS}(L_c, L_e, M(c, e), E_+, E_-).$$

The learning problem is then defined as follows:

Given $L_c, L_e, M(c, e), E_+, E_-$.

Find the version space $VS(L_c, L_e, M(c, e), E_+, E_-)$.

In the sequel, we shall usually fix L_c, L_e , and $M(c, e)$ and write $VS(E_+, E_-)$ or even just VS for short. Version spaces are often considered in terms of *boundary sets* proposed in [45]. They can be defined if the language L_c is *admissible*, i.e., if every chain in the subsumption order has a minimal and a maximal element. In this case,

$$\begin{aligned} GVS(L_c, L_e, M(c, e), E_+, E_-) &:= \text{MIN}_{\sqsubseteq}(VS) := \{c \in VS \mid \neg \exists c_1 \in VS \ c_1 \sqsubset c\}, \\ SVS(L_c, L_e, M(c, e), E_+, E_-) &:= \text{MAX}_{\sqsubseteq}(VS) := \{c \in VS \mid \neg \exists c_1 \in VS \ c \sqsubset c_1\}. \end{aligned}$$

If a version space VS is fixed, we also use notation $G(VS)$ and $S(VS)$ for short. According to [46], the ideal result of learning a target attribute is the case where the version space consists of a single element. Otherwise, the target attribute is said to be *learned partially*.

The elements of the version space can be used as potential classifiers for the target attribute: A classifier $c \in VS$ *classifies* an example positively if c matches e and negatively else. Then, all positive examples are classified positively, all negative examples are classified negatively, and undetermined examples may be classified either way. If it is assumed that E_+ and E_- carry sufficient information about the target attribute, we may expect that an undetermined example is likely to have the target attribute if it is classified positively by a large percentage of the version space (cf. [46]). We say that an example e is α -*classified* (or $\alpha\%$ -classified) if no less than $\alpha \cdot |VS|$ classifiers classify it positively. This means, e.g., that 100%-classification of e takes place if e is matched by all elements of SVS and negative classification of e (0%-classification) takes place if e is not matched by any element of GVS .

5.2 Version Spaces in Terms of Galois Connections

As we showed in [21] the basic properties of general version spaces can easily be expressed with Galois connections, which underlie basic definitions of Formal Concept Analysis [23]. Consider the formal context (E, C, I) , where E is the set of examples containing the disjoint sets of observed positive and negative examples: $E \supseteq E_+ \cup E_-$, $E_+ \cap E_- = \emptyset$, C is the set of classifiers and the relation I corresponds to the matching predicate $M(c, e)$: for $c \in C$, $e \in E$ the relation eIc holds iff $M(c, e) = 1$. The complementary relation, \bar{I} , corresponds to the negation: $e\bar{I}c$ holds iff $M(c, e) = 0$. As shown in [21]

$$VS(E_+, E_-) = E_+^I \cap E_-^{\bar{I}}.$$

This characterization of version spaces implies immediately the property of *merging version spaces*, proved in [30]: For fixed $L_c, L_e, M(c, e)$ and two sets E_{+1}, E_{-1} and E_{+2}, E_{-2} of positive and negative examples one has

$$VS(E_{+1} \cup E_{+2}, E_{-1} \cup E_{-2}) = VS(E_{+1}, E_{-1}) \cap VS(E_{+2}, E_{-2}).$$

This follows from the relation $(A \cup B)' = A' \cap B'$, which holds for a derivation operator $(\cdot)'$ of an arbitrary context.

The classifications produced by classifiers from the version space are characterized as follows. The set of all 100%-classified examples defined by the version space $\text{VS}(E_+, E_-)$ is given by

$$(E_+^I \cap E_-^{\bar{I}})^I.$$

In particular, if one of the following conditions is satisfied, then there cannot be any 100%-classified undetermined example:

1. $E_- = \emptyset$ and $E_+^{II} = E_+$,
2. $(E_+^I \cap E_-^{\bar{I}})^I = E_+$.

The set of examples that are classified positively by at least one element of the version space $\text{VS}(E_+, E_-)$ is given by

$$E \setminus (E_+^I \cap E_-^{\bar{I}})^{\bar{I}}.$$

5.3 Version Spaces for Classifier Semilattices

In the preceding section we showed that the language of FCA and Galois connections is a convenient means for describing version spaces in general case, for unspecified order relation on the set of classifiers. Now we would like to consider a very important special case where the ordered set (C, \leq) of classifiers given in terms of some language L_c makes a meet-semilattice w.r.t. \wedge meet operation. This assumption is quite natural and realistic, e.g., classifiers given as logical formulas form a meet semilattice when the set of these formulas is closed under conjunction. Classifiers given as sets of attributes show the same effect if arbitrary subsets of attribute are allowed as classifiers, too. This also covers the case of attributes with values. In the setting of [46], for example, each attribute takes one of possible values, either constant or “wildcard” $*$, the latter being the shortcut for universal quantification over constant values of this attribute. Examples are given by conjunctions of attribute values. A classifier c matches example e if all attribute values of c that do not coincide with the corresponding values of e are wildcards.

In [21] we proved that in case where the classifiers, ordered by subsumption, form a complete semilattice, the version space is a complete subsemilattice for any sets of examples E_+ and E_- . For the case where the set of classifiers C makes a complete semilattice (C, \sqcap) , we can consider a *pattern structure* $(E, (C, \sqcap), \delta)$, where E is a set (of “examples”), δ is a mapping $\delta : E \rightarrow C$, $\delta(E) := \{\delta(e) \mid e \in E\}$. The subsumption order can be reconstructed from the semilattice operation: $c \sqsubseteq d \iff c \sqcap d = c$.

The version space may be empty, in which case there are no classifiers separating positive examples from negative ones. This happens, e.g., if there is a *hopeless* positive example (an outlier), by which we mean an element $e_+ \in E_+$ having a negative counterpart $e_- \in E_-$ such that every classifier which matches e_+ also matches e_- . An equivalent formulation of the hopelessness of e_+ is that $(e_+)^{\diamond\diamond} \cap E_- \neq \emptyset$.

Theorem 1 *Suppose that the classifiers, ordered by subsumption, form a complete meet-semilattice (C, \sqcap) , and let $(E, (C, \sqcap), \delta)$ denote the corresponding pattern structure. Then the following are equivalent:*

1. *The version space $VS(E_+, E_-)$ is not empty.*
2. *$(E_+)^{\diamond\diamond} \cap E_- = \emptyset$.*
3. *There are no hopeless positive examples and there is a unique minimal positive hypothesis h_{min} .*

In this case, $h_{min} = (E_+)^{\diamond}$, and the version space is a convex set in the lattice of all pattern intents, ordered by subsumption, with maximal element h_{min} .

In case where conditions 1-3 are satisfied, the set of training examples is often referred to as *separable* in machine learning. The theorem gives access to an algorithm for generating the version space. For example, in [21] we use a modification of a standard **Next Closure** [23] algorithm for generating all formal concepts of a formal context to generate the version space as a convex set of the type described in Theorem 1.

According to [23] a subset $A \subseteq M$ can be defined as a *proper premise* of an attribute $m \in M$ if $m \notin A$, $m \in A''$ and for any $A_1 \subset A$ one has $m \notin A_1''$. In particular we can define a *positive proper premise* as a proper premise of the target attribute ω . In [21] we generalized this notion to include the possibility of the unknown value of a target attribute (for an undetermined example): $d \in L_c$ is a *positive proper predictor with respect to examples E_+ , E_- , and E_τ* if the following conditions 1-3 are satisfied:

1. $d^\diamond \subseteq E_+ \cup E_\tau$,
2. $\exists g \in E_+ : g \in d^\diamond$ (or $d^\diamond \cap E_+ \neq \emptyset$),
3. $\forall d_1$ such that $d \sqsubseteq d_1$ and $d \neq d_1$, the relation $d_1^\diamond \not\subseteq E_+ \cup E_\tau$ holds.

In the case where $E_\tau = \emptyset$, satisfaction of condition 2 of the definition follows from condition 1 and a proper predictor is just a *proper premise* [23] of the target attribute. The set of all positive proper predictors for a pattern structure $\Pi = (E, (C, \sqcap), \delta)$ and sets of positive and negative examples E_+ and E_- will be denoted by $PP_+(\Pi, E_+, E_-)$.

By $H_+(\Pi, E_+, E_-)$ we denote the set of positive hypotheses, by $VS(\Pi, E_+, E_-)$ we denote the version space for the pattern structure $\Pi = (E, (C, \sqcap), \delta)$ and sets of positive and negative examples E_+ and E_- . Then the proper predictors and hypotheses are related to the boundaries of the version space as follows [21]:

- (1) $PP_+(\Pi, E_+, E_-) = \text{MAX}_{\sqsubseteq}(\bigcup_{F_+ \subseteq E_+} GVS(\Pi, F_+, E_-))$,
- (2) $H_+(\Pi, E_+, E_-) = \bigcup_{F_+ \subseteq E_+} SVS(\Pi, F_+, E_-)$.

To sum up the relation of concept-based hypotheses with version spaces, we can say the following:

The major drawback of the version spaces where classifiers are defined syntactically is the very likely situation when - in case of too restrictive choice of the classifiers - there is no classifier that matches all positive examples (so-called

“collapse of the version space”). This can easily happen for example when classifiers are just conjunctions of attribute value assignments and “wildcards” $*$, a case mentioned above. In other words: The situation discussed in Theorem 1, which presupposes that there are classifiers that match all positive and no negative examples, is too narrow. If the expressive power is increased syntactically, e.g., by introducing disjunction, then the version space tends to become trivial, while the most specific generalization of positive examples becomes “closer” to or just coincide with the set of positive examples. A syntactical restriction to conjunctions of k -term disjunctions was proposed in [57]. Hypotheses as we defined them in terms of concepts and patterns structures offer another sort of “context-restricted” disjunction: not all disjunctions are possible, but only those of minimal hypotheses (that are equivalent to certain conjunctions of attributes), which express similarities of examples. As for the relation of version spaces (with example descriptions given by conjunctions of attribute values) to decision trees, for any $c \in G(VS)$ there is a decision tree with a path whose set of attributes coincide with c , but in general not all paths of decision trees are in $G(VS)$.

A systematical elaboration of the idea of version spaces in logical terms by means of logical programming is going on in Inductive Logic Programming (ILP, see [47, 48]). Classifiers there are logical formulas with deducibility order on them. Standard inductive operators of ILP, called V - and W -operators, are based on the idea of inverting resolution. The application of these operators can be translated in the language of FCA generally in terms of association rules (partial implications) and for certain cases in terms of implications.

6 Applications of Concept-Based Hypotheses

Starting from early 1980s JSM-hypotheses (equivalent to concept-based hypotheses from Section 2) were used in several applied domains, including biosciences, technical diagnostics, sociology, document dating, spam filtering and so on. Most numerous experiments were carried out in applied pharmacology or Structure-Activity Relationship domain, which deals with predicting biological activity of chemical compounds with known molecular structure. JSM-hypotheses were generated for antitumor [53], antibacterial, antileptous, hepatoprotective [7], plant growth-stimulating, cholesterase-inhibitine, toxic and carcinogenic activities, see review [4]. A freeware system QuDA [25, 26], which incorporates several data mining techniques also presents a possibility of generating JSM-hypotheses. JSM-hypotheses were used for making predictions at two international competitions: that for predictive toxicology [28, 5] and that for spam filtering [10].

6.1 Competition on Predictive Toxicology

The program of a workshop on Predictive Toxicology Challenge (PTC) [28], (at the joint 12th European Conference on Machine Learning and the 5th European Conference on Principles of Knowledge Discovery in Databases) consisted

in a competition of machine learning programs for generation of hypothetical causes of toxicity from positive and negative examples of toxicity. The organizers (Machine Learning groups of the Freiburg University, Oxford University, and University of Wales) together with toxicology experts (US Environmental Protection Agency, US National Institute of Environmental and Health Standards) provided the participants with training and test samples.

The training sample consisted of descriptions of 185 molecular graphs of 409 chemical compounds with indication of whether a compound is toxic or not for a particular sex/species group out of four possible groups: male mice, female mice, male rats and female rats. For each group there were about 120 to 150 positive examples and 190 to 230 negative examples of toxicity with indication of whether a substance is toxic for four sex/species groups: $\{\text{male, female}\} \times \{\text{mice, rats}\}$ (for some groups a substance can be neither a positive nor a negative example because of ambiguous laboratory results). The test sample provided by the Food and Drug Administration (FDA) consisted of 185 substances for which forecasts of toxicity should be made (actually, (non)toxicity of substances was known to organizers). Twelve research groups (world-wide) participated in PTC, each with up to 4 prediction models for every sex/species group.

The competition consisted of the following stages: 1. Encoding of chemical structures in terms of attributes, 2. Generation of classification rules, 3. Prediction by means of classification rules. All results of each stage were made public by the organizers. In particular, encodings of chemical structures made by a participant were made available to all participants. The evaluation was ROC diagrams where each predictive model was represented in a two-dimension space with coordinates related to the rate of (in)correctly predicted toxicity: percentage of substances from the test sample with correctly predicted toxicity (true positive classification rate) and that of incorrectly predicted (false positive classification rate).

The following learning models were used by the participants: structural regression tree (STR) [33] based on combination of statistical methods for constructing regression trees and inductive logic programming (ILP); tree induction for predictive toxicology (TIPT) [3], which is an adaptation of the standard C4.5 algorithm, ILP algorithm PROGOL [47] that realizes inverse entailment for generalizing positive examples w.r.t. a partial domain theory; LRD model based on Distill algorithm [58], which is a combination of the method of Disjunctive Version Spaces (DiVS) [57] with the method of stochastic choice of certain model parameters; the OUCL-2 model used the C4.5 algorithm for construction decision trees, where some attributes were constructed by means of ILP and WARMR methods [32] (the latter is a modernization of the Apriori algorithm [1] by generating DATALOG queries levelwise up to a certain level, choosing those satisfied by a sufficient number of examples); OAI model used a combination of rules generated by C4.5 with Bayes classification followed by voting; LEU3 model based on the Inductive constraint logic (ICL) algorithm [12], which used a mutagenesis theory preconstructed by PROGOL; LEU1 model used an algorithm for inducing decision trees; LEU2 model based on the MACCENT

system with the use of association rules found by WARMR. MACCENT model [11] also uses DATALOG queries, first finding constraints for conditional distributions for the membership to positive and negative classes and then finding the distribution with the use of the maximum entropy principle.

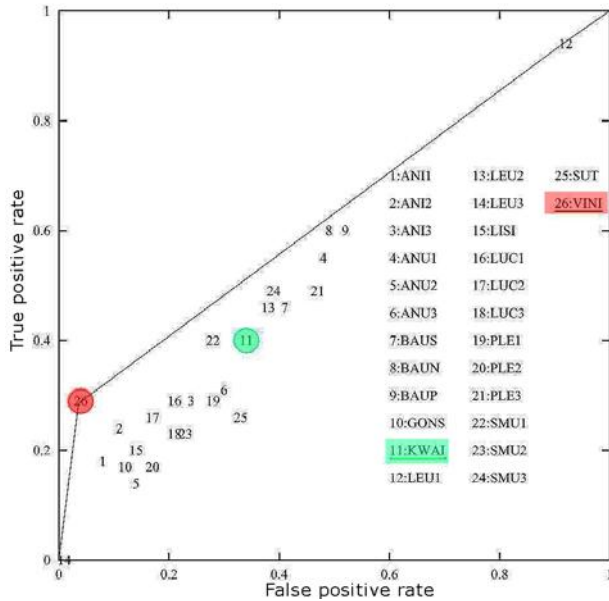


Fig. 4. ROC-diagram for classifications in the female mice group

As measured by ROC diagrams, the performance of the learning program from VINITI (Moscow) [5] based on JSM-hypotheses defined in Section 2 turned out to be Pareto-optimal (ROC diagrams allow for several incomparable “best” results) for all sex/species groups, see e.g. model number 26 in Fig. 4, among all classification rules generated by learning models participating in the competition in terms of the relative number of false and true positive classifications made by hypotheses generated by a learning program (see [28] for details).

6.2 Spam Filtering

The first successful applications of concept-based hypotheses for filtering spam was reported in [15]. In April-May 2003 Technical University Chemnitz, European Knowledge Discovery Network, and PrudSys AG organized the Data Mining Cup (DMC) competition for students specializing in Machine Learning [10]. 514 participants from 199 Universities of 38 countries received training datasets with 8000 e-mail messages some part of which (39%) was qualified as spam (positive examples) and the rest (61%) as nonspam (negative examples). The test dataset contained 11177 messages. Both datasets were described by 833

attributes, including 832 binary ones and one numeric one. The only numeric attribute (ID) reflected the (unique) incoming number of the e-mail within the company that provided the data. Thus no two different e-mails had the same value for this attribute.

The participants were to generate a classifier for distinguishing spam from not spam by using various machine learning techniques. An important condition of the competition was that the error of classifying a nonspam message from the test set as a spam one should not be greater than 1%. There were only 74 participants whose learning models did not exceed this level. The learning models were then ranked according to the rate of incorrect classification of spam messages as nonspam ones.

Seven students from Computer Science and Mathematics Faculties of the Darmstadt University of Technology took part in the competition. Three of the students obtained solutions within the best 20 in the competition. The sixth place (the best among the Darmstadt group) was taken by a solution of F. Hutter, which used “Naive Bayes” approach with boosting and when the confidence level of the classification (i.e., the rate of correct classifications in the training sample itself) was less than 90% utilized concept-based hypotheses with support (i.e., number of examples in the hypothesis extent) ≥ 20 . When hypotheses refused to classify a message from the test set, the model used classifiers based on majority votes among decision trees, the Naive Bayes classifier and a neural network. The sixteenth and seventeenth places in the competition were also taken by the students from Darmstadt. Their models combined concept-based hypotheses, decision trees and Naive Bayes approaches using the majority vote strategy.

During the data preparation stage, the participants from Darmstadt “cleaned off” the ID attribute (the serial number of an e-mail message). It turned out later that the ID attribute, unique for each e-mail, implicitly indicated the time when the e-mail was received: the last 4000 e-mails were spam (obtained on holidays when business correspondence was temporarily canceled). 5 most successful models used this to turn ID into time attribute with few values (roughly, “before holidays” and “during the holidays”). The results obtained with concept-based hypotheses could have been even better if this consideration had been taken into account at the preprocessing stage.

7 Conclusions

The application of the lattice theory and FCA in machine learning shows that the basic notions in lattice-based learning are that of a concept, concept intent (closed itemset), implication, and association rule (partial implication). We presented several machine learning models from the concept lattice viewpoint, including version spaces, decision trees, and JSM- (or concept-based) hypotheses. It is shown that concept-based hypotheses tend to be “more cautious” than those obtained by decision trees. On the other hand, they introduce a kind of restricted disjunction (over a certain subset of concept intents) for purely conjunctive version spaces. A next interesting link between FCA and machine learning will be

relating FCA-based learning with methods of ILP. We also discussed algorithmic problems of concept-based and pattern-based hypotheses, as well as applications of concept-based hypotheses in predictive toxicology and spam filtering.

Acknowledgments

I thank Bernhard Ganter, Peter Grigoriev, Sergei Obiedkov, and Mikhail Samokhin for helpful discussions and attention to this work.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo, Fast Discovery of Association Rules, in *Advances in Knowledge Discovery and Data Mining*, 1996, 307-328.
2. F. Baader and R. Molitor, Building and Structuring Description Logic Knowledge Spaces Using Least Common Subsumers and Concept Analysis, in *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'2000*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 292-305.
3. D. Bahler and D.W. Bristol, The induction of rules for predicting chemical carcinogenesis in rodents, in *Intelligent Systems for Molecular Biology*, L. Hunter, D. Searls, J. Shavlik, Eds., Menlo Park, CA, AAAI/MIT Press, 1993, 29-37.
4. V.G. Blinova, Results of Application of the JSM-method of Hypothesis Generation to Problems of Analyzing the Relation "Structure of a Chemical Compound - Biological Activity," *Autom. Docum. Math. Ling.*, vol. 29, no. 3, pp. 26-33, 1995.
5. V.G. Blinova, D.A. Dobrynin, V.K. Finn, S.O. Kuznetsov, and E.S. Pankratova, Toxicology analysis by means of the JSM-method, *Bioinformatics* 2003, vol. 19, pp. 1201-1207.
6. M. Botta, A. Giordana, L. Saitta, and M. Sebag, Relational Learning as Search in a Critical Region, *Journal of Machine Learning Research*, 2003, **4**, 431-463.
7. A.P. Budunova, V.V. Poroikov, V.G. Blinova, and V.K. Finn, The JSM-method of hypothesis generation: Application for analysis of the relation "Structure - hepatoprotective detoxifying activity", *Nauchno-Tekhnicheskaya Informatsiya*, no. 7, pp.12-15, 1993 [in Russian].
8. C. Carpineto, G. Romano, A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval, *Machine Learning*, 1996, Vol. 24, pp. 95-122.
9. L. Chaudron and N. Maille, Generalized Formal Concept Analysis, in *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'2000*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 357-370.
10. Data Mining Cup (DMC), <http://www.data-mining-cup.de>
11. L. Dehaspe and L. De Raedt, Mining Association Rules in Multiple Relations, *Proc. 7th Int. Workshop on Inductive Logic Programming, LNAI*, vol. 1297, 1997, 125-132.
12. L. De Raedt, W. van Laer, Inductive Constraint Logic, *Proc. 5th Workshop on Algorithmic Learning Theory, LNCS*, vol. 997, 1995, 80-94.
13. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From Data Mining to Knowledge Discovery: An Overview, In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Eds. *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996, 3-33.

14. S. Ferré and O. Ridoux, A Logical Generalization of Formal Concept Analysis, in *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'2000*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000.
15. S. Ferré and O. Ridoux, The Use of Associative Concepts in the Incremental Building of a Logical Context in *Proc. 10th Int. Conf. on Conceptual Structures, ICCS'2002*, U. Priss, D. Corbet, G. Angelova, Eds., Lecture Notes in Artificial Intelligence, **2393**, 2002, 299-313.
16. V.K. Finn, On Machine-Oriented Formalization of Plausible Reasoning in the Style of F. Backon-J. S. Mill, *Semiotika Informatika*, **20** (1983) 35-101 [in Russian].
17. V.K. Finn, Plausible Reasoning in Systems of JSM Type, *Itogi Nauki i Tekhniki, Seriya Informatika*, **15**, 54-101, 1991 [in Russian].
18. J.G. Ganascia, CHARADE: A rule system learning system, In: *Proc. of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, August 23-28 (1987), 345-347, 1987.
19. B. Ganter and S. Kuznetsov, Formalizing Hypotheses with Concepts, *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'00*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 342-356.
20. B. Ganter and S. Kuznetsov, Pattern Structures and Their Projections, *Proc. 9th Int. Conf. on Conceptual Structures, ICCS'01*, G. Stumme and H. Delugach, Eds., Lecture Notes in Artificial Intelligence, **2120** 2001, pp. 129-142.
21. B. Ganter and S.O. Kuznetsov, Hypotheses and Version Spaces, *Proc. 10th Int. Conf. on Conceptual Structures, ICCS'01*, A.de Moor, W. Lex, and B.Ganter, Eds., Lecture Notes in Artificial Intelligence, **2746** 2003, pp. 83-95.
22. B. Ganter and K. Reuter, Finding all closed sets: a general approach, *Order*, **8**, 283-290, 1991.
23. B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer, 1999.
24. M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York, Freeman, 1979.
25. P.A. Grigoriev and S.A. Yevtushenko, Elements of an Agile Discovery Environment, in *Proc. 6th International Conference on Discovery Science (DS 2003)*, Eds. G. Grieser, Y. Tanaka, and A. Yamamoto, Lecture Notes in Artificial Intelligence, **2843**, 309-316, 2003.
26. P.A. Grigoriev, S.A.Yevtushenko, and G.Grieser, *QuDA, a data miner's discovery environment*, Technical Report AIDA 03 06, FG Intellektik, FB Informatik, Technische Universitaet Darmstadt, September 2003, <http://www.intellektik.informatik.tu-darmstadt.de/peter/QuDA.pdf>.
27. C.A. Gunter, T.-H. Ngair, D. Subramanian, The Common Order-Theoretic Structure of Version Spaces and ATMSs, *Artificial Intelligence* **95**, 357-407, 1997.
28. C. Helma, R.D. King, S. Kramer, A. Srinivasan, *Proc. of the Workshop on Predictive Toxicology Challenge* at the 5th Conference on Data Mining and Knowledge Discovery (PKDD'01), Freiburg (Germany), 2001, September 7, <http://www.predictive-toxicology.org/ptc/>
29. J. Hereth, G. Stumme, R. Wille, and U. Wille, Conceptual Knowledge Discovery and Data Analysis, *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'98*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 421-437.
30. H. Hirsh, Generalizing Version Spaces, *Machine Learning* **17**, 5-46, 1994.
31. H. Hirsh, N. Mishra, and L. Pitt, Version Spaces Without Boundary Sets, in *Proc. of the 14th National Conference on Artificial Intelligence (AAAI97)*, AAAI Press/MIT Press, 1997.

32. R.D. King, A. Srinivasan, and L. Dehaspe, WARMR: A Data Mining tool for chemical data, *J. Computer-Aided Molecular Design*, vol. 15, 2001, 173-181.
33. S. Kramer, Structural Regression Trees, *Proc. 13th National Conference on Artificial Intelligence (AAAI-96)*, 812-819, AAAI Press, 1996.
34. S.O. Kuznetsov, Interpretation on Graphs and Complexity Characteristics of a Search for Specific Patterns, *Nauchn. Tekh. Inf., Ser. 2 (Automat. Document. Math. Linguist.)* no. 1, 23-27, 1989.
35. S.O. Kuznetsov, JSM-method as a machine learning method, *Itogi Nauki i Tekhniki, ser. Informatika*, vol. 15, pp.17-50, 1991 [in Russian].
36. S.O. Kuznetsov and V.K. Finn, On a model of learning and classification based on similarity operation, *Obozrenie Prikladnoi i Promyshlennoi Matematiki* **3**, no. 1, 66-90, 1996 [in Russian].
37. S.O. Kuznetsov, Learning of Simple Conceptual Graphs from Positive and Negative Examples. In: J. Zytkow, J. Rauch (eds.), *Proc. Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD'99*, Lecture Notes in Artificial Intelligence, **1704**, pp. 384-392, 1999.
38. S.O. Kuznetsov, On Computing the Size of a Lattice and Related Decision Problems, *Order*, 2001, **18**(4), pp. 313-321.
39. S.O. Kuznetsov, S.A. Obiedkov, Comparing performance of algorithms for generating concept lattices, *J. Exp. Theor. Artif. Intell.*, 2002, vol. 14, nos. 2-3, pp. 189-216.
40. M. Liquiere and J. Sallantin, Structural Machine Learning with Galois Lattice and Graphs, *Proc. Int. Conf. Machine Learning ICML'98*, 1998.
41. D. Maier, *The Theory of Relational Databases*, Comput. Sci. Press, Potomac, MD, 1983.
42. M. Luxenburger, Implications partielles dans un contexte, *Math. Sci. Hum.*, 1991.
43. P. Njiwoua, E. Mefu Nguifo, Forwarding the choice of bias. LEGAL-F: Using Feature Selection to Reduce Complexity of LEGAL, in *Proc. of BENELEARN-97*, 1997, pp. 89-98.
44. T. Mitchell, Version Space: An Approach to Concept Learning, PhD thesis, Stanford University, 1978.
45. T. Mitchell, Generalization as Search, *Artificial Intelligence* **18**, no. 2, 1982.
46. T. Mitchell, *Machine Learning*, The McGraw-Hill Companies, 1997.
47. S. Muggleton, Inverse Entailment and Progol, *New Generation Computing*, Special Issue on Inductive Logic Programming, vol. 13 (3-4), 1995, 245-286.
48. S.-H. Nienhuys-Cheng and R. de Wolf, *Foundations of Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, vol. 1228, 1997.
49. G.D. Oosthuizen and D.R. McGregor, Induction Through Knowledge Normalization, in *Proc. 8th. European Conference on Artificial Intelligence*, Munich, 1988.
50. G.D. Oosthuizen, The use of a lattice in Knowledge Processing, PhD Thesis, University of Strathclyde, Glasgow, 1988.
51. G.D. Oosthuizen, The application of Concept Lattices to Machine Learning, University of Pretoria, Tech. Rep. CSTR 94/01, 1994.
52. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, Efficient Mining of Association Rules Based on Using Closed Itemset Lattices, *J. Inf. Systems*, **24**, 1999, pp. 25-46.
53. D.V. Popov, V.G. Blinova and E.S. Pankratova, Drug design. JSM-Method of hypothesis generation for predicting antitumor activity and toxic effects forecasts with respect to plant products, *Proc. 5th Int. Conf. on Chemistry and Biotechnology of Biologically Active Natural Products*, Varna (Bulgaria), September 18-23, 1989, vol. 2, pp. 437-440.

54. J.R. Quinlan, Induction on Decision Trees, *Machine Learning*, **1**, No. 1, 81-106 (1986).
55. M. Sahami, Learning Classification Rules Using Lattices, *Proc. 8th European Conference on Machine Learning*, N. Lavrac, S. Wrobel, Eds., pp. 343-346, 1995.
56. M. Sebag, Using Constraints to Building Version Spaces, in L. de Raedt and F. Bergadano, eds., *Proc. of the European Conference on Machine Learning (ECML-94)*, pp. 257-271, Springer, 1994.
57. M. Sebag, Delaying the Choice of Bias: A Disjunctive Version Space Approach, in L. Saitta ed., *Proc. of the 13th International Conference on Machine Learning*, pp. 444-452, Morgan Kaufmann, 1996.
58. M. Sebag, C. Rouveroil, Tractable induction and classification in first-order logic via stochastic matching, *Proc. 15th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1997, 888-893.
59. E.N. Smirnov and P.J. Braspenning, Version Space Learning with Instance-Based Boundary Sets, in H. Prade, ed., *Proc. of 13th European Conference on Artificial Intelligence*, J. Wiley, Chichester, 460-464, 1998.
60. A. Srinivasan, S.H. Muggleton, M.J.E. Sternberg, and R.D. King, Theories for mutagenicity: a study in first order and feature-based induction, *Artificial Intelligence*, 1996, **85**, 277-299.
61. G. Stumme, R. Wille, U. Wille, Conceptual Knowledge Discovery in Databases Using Formal Concept Analysis Methods, In: J. Zytkow, M. Quafoufou, ed., *Proc. 2nd European Symposium on PKDD'98. Principles of Data Mining and Knowledge Discovery*, Lecture Notes in Artificial Intelligence, **1510**, Springer 1998, 450-458.
62. R. Wille, Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts, In: *Ordered Sets* (I. Rival, ed.), Reidel, Dordrecht-Boston, 445-470, 1982.

A Comparative Study of FCA-Based Supervised Classification Algorithms

Huaiyu Fu, Huaiguo Fu, Patrik Njiwoua, and Engelbert Mephu Nguifo

CRIL-CNRS FRE2499, Université d'Artois - IUT de Lens
Rue de l'université SP 16, 62307 Lens cedex, France
`{fu,mephu}@cril.univ-artois.fr`

Abstract. Several FCA-based classification algorithms have been proposed, such as GRAND, LEGAL, GALOIS, RULEARNER, CIBLe, and CLNN & CLNB. These classifiers have been compared to standard classification algorithms such as C4.5, Naïve Bayes or IB1. They have never been compared each other in the same platform, except between LEGAL and CIBLe. Here we compare them together both theoretically and experimentally, and also with the standard machine learning algorithm C4.5. Experimental results are discussed.

1 Introduction

Supervised classification is an essential task of machine learning (ML) and data mining (DM). Supervised classification aims to find the rules that predict the class label of an object. Once the data set is fixed, supervised classification process consists of two phases: learning and classification.

In the learning (or training) phase, a classifier (or model) is constructed by analyzing objects (training set) that are described by attributes. The class label is set for every object of the training set. In the classification (or test) phase, the classifier is used to predict the class label of new (or unseen) objects. The set of these new objects is called the test set. In order to get effective classification for the new objects, we should ensure that the accuracy of the classifier is acceptable. So the predictive accuracy of the classifier must be estimated by using different statistics measures [1].

A class C_i means a subset of objects, consisting of all objects that satisfy the designed class condition. Positive examples of the class C_i are the objects that satisfy the class condition. The examples outside the set of the positive examples are negative examples. Once the classes are defined, the classification system should infer the rules that govern the classification i.e. the system should find the description of each class. Ideally all of the positive examples should satisfy the description and none of the negative examples. A rule is said to be correct if its description covers all the positive examples and none of the negative examples.

Many classification algorithms have been developed [2] and widely applied in practice. Each method has its characteristics which can be adapted to certain classification task. If there are many publications on refinement of basic

well-known algorithms, FCA-based classifiers still need to prove their efficiency, especially on the computational aspect. Recent work of Xie and al. [3] shows that they are still many approaches to explore in order to build efficient FCA-based classifiers. Our paper recall the published FCA-based supervised classification algorithms, and compared them w.r.t. both theoretical and practical aspects.

Concept Lattice (CL) is the heart of FCA. CL structure is an effective tool for data analysis, knowledge discovery, ML, information retrieval and document browsing, and more especially for classification and association rules. Theoretical foundation of CL rests on the mathematical lattice theory [4, 5].

Several FCA-based algorithms were proposed for supervised classification task, such as GRAND [6], LEGAL [7], GALOIS [8], RULEARNER [9], CIBLe [10], CLNN & CLNB [3]. These lattice-based algorithms use different strategies for learning and classification. The authors of these algorithms have compared their algorithms with the standard classification algorithms such as C4.5, Naïve Bayes or IB1. But these algorithms have not been compared together in the same platform. So in our work, we present a theoretical comparison of published version of these algorithms, using different criteria of ML systems such as the kind of data, the learning strategy and the classification strategy. We implement in Java some of these algorithms and compare their predictive accuracies as well as their computational costs, on some real data of ML benchmark at UCI repository. Experimental results are discussed.

Considering that the reader is familiar with the basic notions of FCA, the rest of this paper is organized as follows: the lattice-based classification algorithms are introduced in the next section, and compared theoretically. In section 3, the experimental comparison of these algorithms is shown.

2 Theoretical Comparison of FCA-Based Classification Algorithms

This comparison is summarized in table 2.

2.1 Type of Data and Number of Classes

A classification system can deal with different types of learning data such as binary data, valued attribute, numeric and symbolic data, etc. While GRAND and LEGAL can analyze only binary data, GALOIS and RULEARNER deal with valued attribute data. Both numeric and symbolic data can be treated by CIBLe and CLNN & CLNB.

In addition, a classification system must be able to analyse datasets with different number of classes. GRAND, GALOIS, CIBLe and CLNN & CLNB can classify datasets containing more than two classes, LEGAL and RULEARNER can classify datasets with two classes (positive and negative examples of one concept). The author of RULEARNER points out that RULARNER can deal with more than two classes, but only the algorithm for two classes is given.

2.2 The Learning Strategies

Lattice Construction. Many algorithms for generating CL have recently been proposed. All the algorithms can be divided into two categories: incremental and non-incremental algorithms. The difference between incremental and non-incremental consists in how the algorithm treats new examples. The incremental algorithms update the local structure of the lattice, while the non-incremental algorithms build the whole lattice structure. Besides, non-incremental algorithms typically apply two strategies: top-down or bottom-up. Top-down strategy begins from the maximal extent to the minimal one to construct CL, bottom-up begins from the minimal extent to the maximal one. Some algorithms may combine certain parameters to build a partial lattice. In these classification systems, GRAND, GALOIS and RULEARNER create incrementally a complete lattice during learning, while LEGAL, CIBLE and CLNN & CLNB build non-incrementally a partial lattice from learning examples.

GRAND constructs incrementally an entire CL with Oosthuizen's algorithm [6] to explicitly represent the concept search space during learning. The first example with all its attributes and class label is inserted in the lattice, for every new example with its class label, if it has the common attributes with those nodes which already exist in the lattice, then the nodes representing their common attributes are created.

LEGAL applies two learning parameters to modify Bordat's algorithm [11] to build a join-semi lattice based on learning examples. The learning algorithm proceeds in a top-down manner. Examples of the initial context are divided into two sets: positive examples and negative examples. The algorithm begins from the most general node (O, \emptyset) . For each new node, the algorithm constructs all its sub-nodes and then retrieves valid nodes by learning parameters, the valid nodes are inserted in the lattice. If no valid node exists, the algorithm stops. The valid nodes are the nodes verified by a great number of positive examples.

GALOIS uses an incremental algorithm to generate an entire CL in the learning phase. RULEARNER uses directly GRAND method to construct the CL.

CIBLE modifies Bordat's algorithm [11] to build a join-semi lattice only from positive examples. The algorithm can retain the nodes in the lattice with a heuristic selection function, and a parameter which specifies the lattice height.

CLNB [3] (Concept Lattice Naïve Bayes) and CLNN [3] (Concept Lattice Nearest Neighbor) build a partial lattice in a top-down manner, which respectively integrates the Naïve Bayes (NB) base classifier and the Nearest Neighbor (NN) base classifier into nodes in the lattice to form a composite classifier. Every node in the lattice verifies three constraints (Support, Precision, Reject) in order to reduce search space.

Concept Selection and Design Method. Concept selection consists in finding a generalization of a set of examples and their descriptions. Classification system can use the selected (or relevant) concepts to form rules, and then predict new examples through these rules.

GRAND selects maximally complete concepts by traversing the downward closure of each class-node. A maximally complete concept is the meet of the set of attribute-nodes of a class. LEGAL, GALOIS and RULEARNER search for coherent concepts that can be also maximal. Coherent concepts are the concepts verified by examples of one class only. A coherent concept is maximal if it has no super concept which is coherent. For CIBLE, CLNN and CLNB, the strategy for concept selection is to obtain relevant concepts, which are concepts that satisfy certain constraint. CIBLE uses lattice level and different selection measures (entropy function, improved entropy function, Laplace's formula). CLNN & CLNB use three constraints (Support, Precision, Reject) to obtain concepts.

Many classification systems integrate different inference types or/and computational paradigms. Induction is one of the inference types, and is used to synthesize information based on a set of observations into a general hypothesis. GRAND, LEGAL, GALOIS and RULEARNER use an inductive method. While CIBLE and CLNN combine induction and k-NN, CLNB combines induction with NB.

Learned Knowledge. Classification systems can take various ways to represent learned knowledge such as classification rules, relevant concepts and training sets. Classification rules can be considered as predictive rules where the antecedent contains the corresponding attributes, the consequent contains the class.

GRAND builds a set of rules, CLNN & CLNB generate a set of meta-rules, LEGAL and GALOIS construct relevant concepts, RULEARNER induces ordered rules set or unordered rules set to represent learned knowledge, and CIBLE generates pertinent concepts and uses training set for the classification purpose.

2.3 The Classification Strategies

In the classification phase, every classification system selects appropriate strategies to predict a class for a new example. These strategies include majority vote, similarity computation, deduction and other methods.

GRAND applies the most specific rules to classify new examples. A majority vote may be adopted. The author suggests to choose the three best rules (the most specific rules for new examples) to vote. The most specific rules are generated from the concepts which contain the largest number of examples and the smallest number of attributes.

LEGAL uses majority vote in the set of pertinent regularity to classify new examples. It provides two parameters which respectively represent the justification and refutation criteria.

GALOIS computes the similarity between a new example and the set of concepts which are coherent and maximal. The similarity between the new example and the concepts is the number of attributes of the concepts which are verified by the new example. Then GALOIS assigns the most similar class to the new example. Majority vote strategy may also be applied.

RULEARNER selects the rules which respect the order of the antecedents (from general to specific) to classify new examples. The class of the first verified rule is assigned to the new example.

CIBLe uses K-NN to predict a class of a new example. The following distance measures are considered: Mahalanobis, Manhattan and Euclidian.

CLNB & CLNN applies verified rules and voting strategy to predict a class of a new example. The verified rules are the set of rules that are activated by the new example.

3 Experimental Comparison

Here we show the experimental comparisons of four methods in the same Java platform: GRAND, LEGAL, GALOIS, RULEARNER. CIBLe is written in C++, we do not have a Java code for CLNN & CLNB. But we expect to complete our implementation of these systems in future work. In this experimental comparison of four methods, test data sets are selected from the UCI repository of ML databases [12]. A brief description of the data is presented in table 1.

Table 1. Datasets used in the comparison.

Database #	Attribute #	Example #	Class #
Shuttle(SH)	6	15	2
Monk-1(M1)	6	432	2
Monk-2(M2)	6	432	2
Monk-3(M3)	6	432	2
Voting(VO)	16	435	2
Balance(BA)	4	625	3
Lenses(LE)	4	24	3
Hayes-Roth(HR)	4	132	3
Lung cancer(LC)	56	32	3
Post operative(PO)	8	90	3
Zoo(ZO)	17	101	7

Classification accuracy can be used to evaluate the performance of classification methods. It is the percentage of the examples which are correctly classified in the test set and can be measured by splitting the data sets into a training set and a test set. In our experiments, a four-fold cross-validation is applied to measure the classification accuracy of the four methods. The algorithms are run on a Pentium III 900 Mhz machine with 256M RAM.

The table 3 separately lists the average classification accuracies, standard deviations and running times of the four methods and C4.5 over the datasets. The classification performance of the four methods can be divided into two groups. GRAND, GALOIS can classify multi-class datasets and LEGAL, RULEARNER can only classify two-class datasets. The statistical analysis on the two-class datasets for the four methods shows that:

- RULEARNER produces the highest average classification accuracies and LEGAL obtains the best running time among the four methods.
- RULEARNER and LEGAL perform better than C4.5 on average classification accuracies, but C4.5 significantly outperforms the four methods in running times.
- RULEARNER, LEGAL and C4.5 outperform GRAND and GALOIS in average classification accuracies.

The comparison between RULEARNER and LEGAL shows that:

- RULEARNER has slightly better performance than LEGAL in average classification accuracies, but the difference is less significant. RULEARNER on 3 datasets (SH, M1, M2) takes advantage on LEGAL, LEGAL on the remaining datasets is better than RULEARNER. LEGAL in running times outperforms RULEARNER.

The experimental comparison of GRAND and GALOIS shows that:

- For two-class datasets, GALOIS performs slightly better than GRAND in average classification accuracies. GALOIS takes advantage on 3 datasets (M1, M2, M3), GRAND has better performance than GALOIS in the remaining datasets.
- For multi-class datasets, GRAND produces slightly better average classification accuracies than GALOIS. but the difference is less significant. Compared to GALOIS, GRAND takes advantage on 4 datasets (BA, LC, PO, ZO), GALOIS has better performance than GRAND in the other one dataset.
- GRAND has better running times than GALOIS for two-class and multi-class datasets.

Why LEGAL performs the best running time? In learning phase, LEGAL modifies the Bordat algorithm to build a concept lattice, Bordat algorithm performs well for construction of diagram graph on data with average density, furthermore, LEGAL only generates a partial lattice by learning parameters. Consequently, LEGAL can obtain the best running time for classification tasks compared to other three methods.

4 Conclusion

In this paper, we have compared some lattice-based classification methods and implemented four (GRAND, LEGAL, GALOIS, RULEARNER) of them in the same platform Java. These lattice-based algorithms use different strategies for learning and classification.

Among these systems, two different approaches are applied to generate the classifiers. The first approach is that the system constructs an entire CL before generating classifier. GRAND, GALOIS and RULEARNER adopt this approach to construct incrementally the CL. LEGAL, CIBLE and CLNN & CLNB adopt another approach which apply some heuristic techniques only to create those useful concepts without constructing the entire concept lattice. The approach reduces the searched space and increases the efficiency of systems. In addition, LEGAL, CIBLE and CLNN & CLNB combine the inductive approach and other classification techniques such as K-NN and NB.

We select some real data of ML benchmark at UCI repository to test these systems. Experimental results show that some of these systems have good performance for classification tasks.

We will analyse the data context which are favorable to FCA-based classifiers compared to well-known classifiers.

Table 2. Description of lattice-based algorithms. (Car&R: Carpineto, C. and Romano).

Classification System		GRAND	LEGAL	GALOIS	RULEARNER		CIBLe	CLNN&CLNB
Type of Data		Bin	Bin	Attr/Val	Attr/Val	multi	Symb.Num.	Symb.Num.
Learnings	Classes	multi	2	multi	2 or multi	multi	multi	multi
	Type	Complete	Partial	Complete	Complete	Complete	Partial	Partial
	Algorithm	Oosthuizen	Bordat	Car&Ro	Oosthuizen	Bordat	Top-down	
	Lattice	Incremental	Yes	No	Yes	No	No	No
	Concept selection	Maximality	Coherence, Maximality	Coherence, Maximality	Support Maximality	Lattice level Entropy	Support, Precision, reject	
Method		Complete	Maximality	Induction	Induction	Induction + K-NN	Induction + K-NN or NB	
Knowledge Learning		Set of rules	Relevant set Concepts	Relevant set Concepts	Ordered set of rules or unordered set of rules	Relevant concepts	Set of rules	
Classification		Vote	Vote	Similarity or Vote	General rule	K-NN	Verified rule +Vote	
Date of Publication		1988	1990	1993	1995	1999	2002	

Table 3. Percentage accuracies and running times(in seconds) of four learning systems.

Database	GALOIS			GRAND			RULEARNER			LEGAL			C4.5		
	Accuracy	Time		Accuracy	Time		Accuracy	Time		Accuracy	Time		Accuracy	Time	
Shuttle	62.70 ± 1.73	0.18		69.35 ± 2.51	0.20		70.31 ± 2.13	0.16		60.39 ± 1.16	0.08		63.72 ± 1.06	0.00	
Monk-1	88.71 ± 1.04	293.86		82.42 ± 2.56	257.68		97.18 ± 0.94	252.74		96.57 ± 0.53	8.27		75.67 ± 0.76	0.82	
Monk-2	58.39 ± 5.56	295.37		53.24 ± 6.83	259.76		73.35 ± 1.17	253.18		72.36 ± 1.56	8.75		66.29 ± 0.56	0.81	
Monk-3	90.38 ± 2.61	287.32		85.65 ± 3.06	250.09		90.28 ± 2.86	260.92		91.68 ± 1.92	9.21		93.46 ± 0.34	0.90	
Voting	80.46 ± 1.33	2564.29		88.35 ± 2.14	2345.71		85.72 ± 2.04	2390.56		92.37 ± 1.76	102.43		90.21 ± 1.86	0.86	
Average 1	76.13 ± 1.94	684.67		75.80 ± 2.21	622.87		83.37 ± 1.74	631.63		82.67 ± 1.39	25.75		80.67 ± 0.92	0.68	
Balance	68.28 ± 4.35	250.22		72.36 ± 3.24	286.37		/	/		/	/		77.46 ± 1.13	0.73	
Lenses	70.89 ± 6.11	0.65		66.78 ± 9.05	1.08		/	/		/	/		80.74 ± 0.16	0.00	
Hayes-Roth	68.01 ± 4.98	7.41		65.51 ± 4.35	5.20		/	/		/	/		62.35 ± 2.03	0.02	
Lung cancer	42.24 ± 2.62	659.79		47.31 ± 3.57	356.52		/	/		/	/		50.58 ± 2.16	0.15	
Post operative	52.94 ± 1.34	29.50		60.23 ± 2.41	27.93		/	/		/	/		61.43 ± 0.72	0.08	
Zoo	89.48 ± 0.82	10.52		91.48 ± 0.74	8.89		/	/		/	/		91.09 ± 0.83	0.07	
Average 2	65.31 ± 3.37	159.68		66.45 ± 3.89	114.33		/	/		/	/		70.58 ± 1.17	0.18	
Average 3	70.25 ± 2.71	398.31		71.15 ± 3.12	345.48		/	/		/	/		75.16 ± 1.05	0.40	

Average1 is for two-class datasets
Average2 is for multi-class datasets
Average3 is for all datasets

Acknowledgements

We are grateful to anonymous reviewers for helpful comments and to Guy Schoonheere for english proofreading. This research benefits from the support of IUT de Lens, Université d'Artois, CNRS and the region Nord/Pas de calais.

References

1. Dietterich, T.: Approximate statistical tests for comparing supervised classification learning algorithms. Research report, Computer Science Dept., Oregon State University (1997)
2. Eklund, P.W.: A performance survey of public domain supervised machine learning algorithms. Technical report, The University of Wollongong, Australia (2002)
3. Xie, Z., Hsu, W., Liu, Z., Lee, M.: Concept lattice based composite classifiers for high predictability. *Journal of Experimental and Theoretical Artificial Intelligence* **14** (2002) 143–156
4. Birkhoff, G.: *Lattice Theory*. 3rd edn. American Mathematical Society, Providence, RI (1967)
5. Ganter, B., Wille, R.: *Formal Concept Analysis. Mathematical Foundations*. Springer (1999)
6. Oosthuizen, G.: The application of concept lattices to machine learning. Technical Report CSTR 94/01, Department of Computer Science, University of Pretoria, Pretoria, South Africa (1994)
7. Liquière, M., Mephu Nguifo, E.: LEGAL: LEarning with GALois lattice. In: *Actes des Journées Françaises sur l'Apprentissage (JFA)*, Lannion, France (1990) 93–113
8. Carpineto, C., Romano, G.: Galois: An order-theoretic approach to conceptual clustering. In: *Proceedings of ICML'93*, Amherst (1993) 33–40
9. Sahami, M.: Learning Classification Rules Using Lattices. In: *Proceedings of ECML'95*, Heraclion, Crete, Greece, Nada Lavrac and Stefan Wrobel eds. (1995) 343–346
10. Njiwoua, P., Mephu Nguifo, E.: Améliorer l'apprentissage à partir d'instances grâce à l'induction de concepts: le système cible. *Revue d'Intelligence Artificielle (RIA)* **13** (1999) 413–440
11. Bordat, J.: Calcul pratique du treillis de galois d'une correspondance. *Mathématiques, Informatiques et Sciences Humaines* **24** (1986) 31–47
12. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998) <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

Modelling Tacit Knowledge via Questionnaire Data

Peter Busch and Debbie Richards

Department of Computing
Macquarie University, Sydney, Australia
{busch,richards}@ics.mq.edu.au

Abstract. The transfer of tacit knowledge is important in ensuring that an organisations most valuable assets do not walk out the door. While much controversy surrounds the definition of tacit knowledge and whether it can be captured, in this paper we follow a psychological approach based on the work of Sternberg at Yale that seeks to measure tacit knowledge via the capture of responses to work-place scenarios. Focusing on the information technology (IT) work-place, we have developed a tacit knowledge inventory which forms part of a questionnaire given to experts and non-experts in three separate IT organisations. In psychology, descriptive statistics are typically used to analyse the responses. We have chosen a more qualitative and visual approach and have used formal concept analysis (FCA) for data analysis that better suits our small sample size. Using FCA we were able to identify participants that responded similarly to the peer-identified experts. In this way the organisation is alerted to the important role these individuals potentially play.

1 Introduction

Tacit knowledge is that vast store of experiences, technical know - how, skill sets and wisdom that permits us to function from a basic survival level, to interacting in our complex knowledge rich societies, that over time largely becomes codified. It is “knowledge that resides in the minds of the people in an organisation but has not been put in structured, document-based form” (Davenport, De Long and Beers 1998). Tacit knowledge has also been likened to intuition, but in reality intuition is more likely ‘used to access tacit knowledge’ (Brockman and Anthony 1998). We believe much tacit knowledge is articulable, whereas intuition is less so, lying more within the subconscious spectrum. There is naturally much more to it than this.

Although Fleck (1997) ascribes tacit knowledge at the whole organisational sense as being a form of meta or cultural knowledge, the purpose of our work is to examine tacit knowledge in individuals and more particularly the diffusion of such knowledge through the organisation. This paper reports research that extends the techniques of Sternberg (1995) by developing and administering a tacit knowledge inventory to IT practitioners; using Formal Concept Analysis

(FCA) (Ganter and Wille 1999, Wille 1982, 1992) to model the data captured; and Social Network Analysis (SNA) (Scott 1991) to map the tacit knowledge flows between participants. In short, we elicit both the ethical and realistic responses (as a pilot study revealed that a participants' actual response often differed from what they believed to be most appropriate) from domain experts and non-experts to a number of typical IT scenarios. The responses, together with biographical data, are captured via a questionnaire. The results are visualized as a lattice permitting an alternative to the descriptive statistics normally produced as a result of questionnaire processing. As a final step, the communication patterns and relationships between respondents are modeled using SNA to determine if any bottlenecks exist. For space, we do not discuss SNA or its use in this paper.

In the next section we introduce the application domain. In section 3 we describe our use of FCA. In section 4 we show how FCA has allowed us to identify individuals behaving similar to the peer-nominated experts. Our conclusions and future directions are given in Section 5.

2 The Application Domain - IT Tacit Knowledge

A study has been conducted in three IT organisations of varying sizes and structure and covering the private and public sectors, which we refer to as organisations X, Y and Z¹. The questionnaire included three parts: biographical data, SNA-related questions requesting the frequency and nature of interactions with others in the organisation and a final part with scenarios and various answer options to consider. We have used FCA to model the biographical data and scenario responses.

The example in Figure 1 shows one of the scenarios randomly assigned to participants. Only answer option 12 for this scenario is shown in the Figure. Unlike the biographical data, the scenario section deals with one dimensional ordinal data. For example, we ask only that respondents select a value from extremely bad through to extremely good. We see an example of a Likert scale in Figure 2. Scenarios are randomly but equally distributed to participants so that a range of scenarios are covered with each individual only needing to respond to six scenarios. Tacit knowledge is measured and identified by determining the responses of those identified as experts by their peers in the survey. The biographical data allows profiles of experts vs non-experts to be developed.

3 Modeling Questionnaire Data with Formal Concept Analysis

Modeling the scenario responses using FCA permits patterns in the results to be visualized. Such an approach is considered useful particularly if questionnaire

¹ The number of respondents involved in the study were 108, 7 and 16 in organisation X, Y and Z, respectively. The data collected is confidential and thus unavailable for distribution.

Scenario 3

You as a team leader are responsible for implementing a payroll system for another branch within the parent organisation. Although you are expected to do the bulk of the work (55%), you do have five other colleagues able to help as you so desire. The project should take 12 months in total to complete.

You have undertaken some of the initial systems design work largely yourself for the past couple of months, and you now require your colleagues to further help you with the next stage which is mainly that of coding.

You are comfortable with hierarchy, however some of your team members are not. You delegate some tasks to subordinates within your team. One of the team members who specialises in programming has been allocated some software specification work, but would prefer really just to be programming. This person has performed well on coding related tasks in the past, but at this point in time lacks project management skills which would prevent him from becoming an effective team leader. Nevertheless you feel that the person should at least do some of the software specification work.

Rate each of the following responses in relation to the given scenario. It is advisable to read all of the responses before replying.

Answer 12 Give him fewer but more specific tasks to do, because it is simply not worth the effort to argue with him. Besides his skills in coding mean that he will be able to effectively contribute here, but then you can be rid of him, to concentrate on testing with other team members of your choice

Fig. 1. Illustrating Sample Scenario 3 with Answer Option 12.

Choose one:	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Extremely Bad		Neither Good nor Bad			Extremely Good	

Fig. 2. Likert scale (of ordinal type) used in the tacit knowledge testing process.

sample sizes are small, as the quantitative data may not be substantial enough to permit valid statistical conclusions to be drawn.

Assuming the reader possesses at least basic familiarity with FCA, we do not introduce it here. Following Wille (1982) we use the terms: (G)egenstande for the objects, (M)erkmale for the attributes, and the relationship between them: (I)deale. We refer to the combination of (G,M,I) as a formal context. A binary K(ontext) (in German) may be formally expressed thus:

$K := (G, M, I)$

A multivalued context may be expressed a quadtuple:

$K := (G, M, W, I)$ and $I \subseteq G \times M \times W$

where the relationship I is a subset of the combined components of Objects (G), Attributes (M) and merksmaleWerte (W) (Attribute-values). In short, “the aim of FCA is to find (formal) concepts and construct the hierarchy between them. To achieve this aim, the many valued context must be transformed into a single valued one by so - called conceptual scaling” (Ganter and Wille 1989 in Bartel and Brüggemann 1998 :24). Along the lines of Kollewe (1989), the

FCA approach is able to process questionnaire data. We translate the multi-valued questionnaire responses into a formal context, a binary cross table which may be constructed, where: K = Formal table with its corresponding data; G = The participant; M = The responses; W = The value of the response; I = Relationship between the responses, their values and the participants.

Kollewe's (1989) use of questionnaire data used the units of questioning as the objects (G), the questions as the attributes (M) and the answers to the questions as the attribute-values (W). In a pilot study (Richards and Busch 2000), it was decided to represent the data differently as it seemed more intuitive. We regarded the participant as the object (G) that has a number of features (M) such as age and position in addition to a set of responses and their values. This approach made data entry and validation easier as there was a one-to-one correspondence between the survey returned and the participant. We begin to implement a crosstable using the Anaconda(software, where the rows are embedded Structured Query Language statements that in turn access the data from a relational database.

We can convert the Likert scale in Figure 2 into a crosstable to provide a visual representation of the scale that will be used in the display of the data. The conversion of likert-scale data to a crosstable has been previously performed by Spangenberg and Wolf (1988). That study used the repertory grid approach to elicit the responses of anorexia nervosa patients to various people in their lives based on a number of bipolar scales. For example, one scale used in the study was a six-point likert scale with open-minded at one end and reserved at the other end. For that scale, if a score of 1 or 2 was given for an individual, the open-minded attribute would be marked with a cross. If a score of 5 or 6 was given, the reserved attribute would be checked. A score of 3 or 4 received no crosses and could be interpreted from the lattice by that individual being neither open-minded or reserved. Our work differs in that we are not concerned with repertory grid data and, as shown in Table 1, we do not handle multi-valued responses in such a simplistic way.

Using the complementary Toscana(software, which takes the Anaconda(data, one is able to eventually construct the complete concept lattice as illustrated in Figure 3. In this instance we are able to visualise the keys (in the relational database sense) of individuals who have chosen particular values for an actual tacit knowledge answer option.

In the development of suitable scales for each of our data elements, we considered possible uses of the subsumption relation. Taking the biographical question, 'please select the highest formal qualification (or equivalent) you have obtained' as an example (Figure 4), we can create a scale which captures the subsumption between each of the qualifications. The scale developed will depend on subjective judgement and in this particular example vary across cultures and societies. For example, in Germany, academic paths tend to be firmly set with less opportunity for 'mind-changing' in later life than in Australia which is more egalitarian (Spillane 1983).

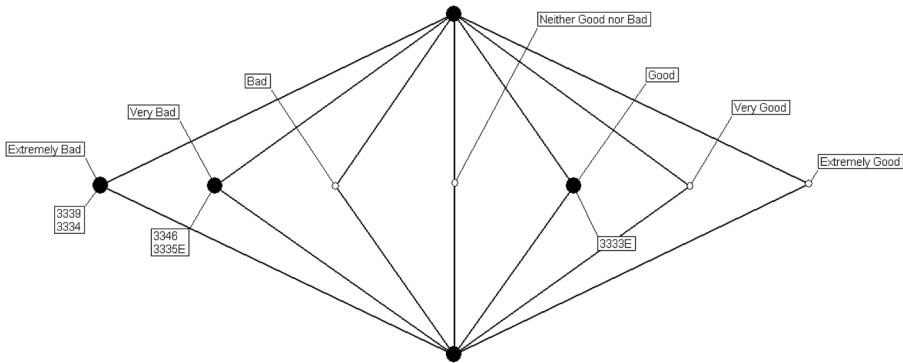


Fig. 3. Illustrating the concept lattice with data included from the database table. Note how the diagram shows results for the values Extremely Bad (2 respondents), Very Bad (2 respondents) and Good (1 respondent).

12. Please select the HIGHEST formal qualification (or equivalent) you have obtained

Fig. 4. Illustrating a biographical question, in this instance highest formal qualification obtained.

4 The Identification of Expert Non-experts

Having collected the survey data and developed the scales to be used in Toscana we began to explore a number of questions we hoped the data would answer. Many of the questions revolved around the responses of experts vs non-experts. Following Sternberg, in our approach, experts were selected by their peers from the list of participants in the online survey. The question involving identification of experts was not specific to a particular scenario. Those not identified as experts became the control or normal group. 32, 5 and 5 individuals from organisations X, Y, Z were nominated as experts by their colleagues. One question of particular interest was whether we could identify participants that were 'hiding their light under a bushel' which would reduce the likelihood of tacit knowledge being transferred by them to other workers. To answer this question we wanted to determine if there were individuals who behaved like experts but had not been

designated as an expert. This new group of people would be the expert non-experts. The ramification for identifying expert non-experts is that this sample group are likely to constitute employees who also have high levels of tacit aka 'managerial' knowledge. This is important to remember when we examine the interactions of staff when looking at the results of the Social Network Analysis.

The creation of a formal concept lattice for each scenario answer, showing the ethical and realistic response of both experts and non-experts, meant that a profile could be created of personnel who had answered close to that of experts. Let us examine this process in closer detail using the sample scenario and answer option given in Figure 3. For answer 12 to scenario 3, initial descriptive statistics revealed that the 17 experts were inclined to be a little more negative or pessimistic (ethically) (mean of 2.5) when dealing with this answer, than the 23 non-experts (mean of 3.2) on the whole. Likewise the experts were also marginally more negative realistically (mean of 3.6), than the non-experts (mean of 3.8). Examining the median values however we note that experts were even more negative ethically (2.0, or Very Bad) than the non-experts (3.0 or Bad). Whilst realistically both groups were actually fairly non-committal (4.0 or Neither Good nor Bad).

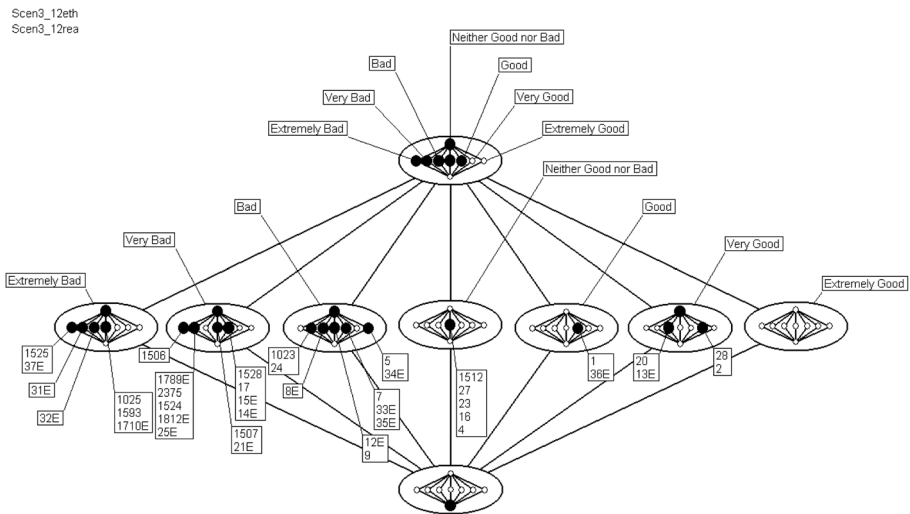


Fig. 5. Illustrating formal concept lattice for Scenario 3, answer 12.

While descriptive statistics allowed us to measure the sensitivity between the expert population and the non-expert population, to determine if there were any statistical differences between the responses of the two populations we performed a Wilcoxon test. Due to the sample size we could not conduct a test for each scenario and answer option but needed to use all the data. We found no statistically significant difference between the expert and normal groups. Given that via FCA analysis we identified 29 individuals in the non expert group as

behaving like experts, this result is not surprising. In fact, we found that once we added the expert non-expert group to the expert group the Wilcoxon test did produce a statistically significant difference for the ethical responses. At this stage we reserve judgement on the value of statistical methods for analyzing this data until we have performed some more sophisticated techniques such as analysis of variance or cluster analysis. We show next how via FCA we were able to perform more fine-grained analysis at the individual level to identify meaningful differences between the expert and non-experts.

Lattices such as that in Figure 5 were used to identify expert non-experts. The reader is reminded that the lattice is read starting from the outer ellipses (ethical values) through to the inner lattice structures (realistic values). Thus in Figure 5 we can already visualise that answers are inclined somewhat toward the negative end of the spectrum for this answer. We can also see that some personnel (1528, 17, 15E, 14E, 7, 33E, 35E, 5, 34E) whilst ethically feeling negative about this particular answer for dealing with scenario 3, nevertheless feel positive about the answer from a realistic standpoint. Alternatively we may note that individuals 20 and 13E feel positive ethically about this answer, but negative about it realistically. Generally speaking however the ethical feeling tends to be more negative than the realistic one.

The point of this exercise is that in examining each formal concept lattice for each answer option for each scenario, we are slowly able to build up a picture for how non-experts have answered relative to that of experts. For example in Figure 5 above we can see that 1525, 1025, 1583, 2375, 1524, 1528, 17, 1507, 9, 7, 5, 1 and 20 have answered the same way as experts. By noting this similarity for all 125 answers (examining all concept lattices) non-experts can be listed in descending order of similarity with expert answer responses. For Organisation X, an extra 25 personnel were able to be identified who consistently scored close to that of experts using the aforementioned technique, without necessarily being identified by their peers as being experts. Examining the closeness of the scores in descending order, it was decided the top 32 group. These were individuals 1, 2, 4, 5, 7, 9, 16, 17, 20, 23, 24, 27, 28, 1023, 1507, 1521, 1524, 1527, 1534, 1536, 1538, 1543, 1778, 1792, 1796.

When dealing with organisation Z, there were 4 other individuals who obtained scores close to that recorded by experts. Once again we note that at least two of the individuals are of non-English speaking origin. Participant 3334 decided not to provide any biographical details, 3339 declined to provide gender. Note once again bachelors degrees and high school certificate as being the highest formal qualifications. As there were only two individuals (out of 7 participants) who were not actually identified as experts in Organisation Y, the same technique was not adopted.

5 Conclusion

Whereas many researchers in the knowledge management field attempt to focus on the tacit component, few means actually exist to measure this type of knowledge. Sternberg's Yale University based approach could be said to be the most

practical because of its applied nature. Thus in our approach we have developed a Sternberg-like workplace-based inventory for the IT-field where employees are asked to make decisions as to how they would handle a soft knowledge situation from an ethical and realistic perspective. Part of the Sternberg approach to tacit knowledge measurement is the identification of experts within the organisation by the participants. We have extended Sternberg's approach by incorporating techniques from FCA and SNA. As demonstrated in this paper, we use FCA to model the questionnaire responses and to further identify individuals (who became the expert non-expert group) that behaved similarly to the peer-selected experts. FCA thus provides a qualitative modeling alternative to the typical quantitative modeling conducted on questionnaire data using statistical methods. While not presented, using SNA we can determine the likelihood of tacit knowledge flowing from the experts and expert non-experts to others within the organisation to minimize the loss or wastage of knowledge resources.

References

1. Bartel, H-G., Brüggemann, R., (1998) "Applications of formal concept analysis to structure - activity relationships" *Fresenius journal of analytic chemistry* 361(1) May 1st. :23 - 28
2. Brockmann, E., Anthony, W., (1998) "The influence of tacit knowledge and collective mind on strategic planning" *Journal of Managerial Issues* Pittsburg; Summer
3. Davenport, T., De Long, D., Beers, M., (1998) "Successful knowledge management projects" *Sloan Management Review* Winter (from ABI Proquest)
4. Fleck, J., (1997) "Contingent knowledge and technology development" *Technology Analysis and Strategic Management* December (From ABI Proquest)
5. Ganter, R., Wille, R., (1999) *Formal concept analysis: Mathematical foundations* Springer - Verlag Berlin Germany
6. Kollewe, W., (1989) "Evaluation of a survey with methods of formal concept analysis" in *Conceptual and numerical analysis of data: Proceedings of the 13th conference of the Gesellschaft für Klassifikation e. V. University of Augsburg*, April 10 - 12, 1989 Springer -Verlag Berlin Germany :123 - 134
7. Richards, D., Busch, P., (2000) "Measuring, formalising and modeling Tacit Knowledge" *International Congress on Intelligent Systems and Applications (ISA2000)* December 12 - 15 2000
8. Scott, J., (1991) *Social Network Analysis: A handbook* Sage Publications London U.K.
9. Spangenberg, N., Wolff, K.E. (1988) "Conceptual grid evaluation", In *Classification and related methods of data analysis*. Spangenberg, Norbert and Karl Erich Wolff, Eds., Amsterdam, North-Holland: 577-580.
10. Spillane, R., (1983) *Stress at work: A review of Australian research* Department of Psychology University of Stockholm Report Number 35
11. Sternberg, R., (1995) "Theory and management of tacit knowledge as a part of practical intelligence" *Zeitschrift für Psychologie* 203(4) :319 - 334
12. Wille, R., (1982) "Restructuring lattice theory: an approach based on hierarchies of concepts". In Reidel, D. *Ordered Sets*, Dordrecht, 1982, pp. 445-470.
13. Wille, R., (1992) *Concept lattices and conceptual knowledge*. *Computers and Mathematics with Applications* **23** (1992) 493-522.

Predicate Invention and the Revision of First-Order Concept Lattices

Michael Bain

School of Computer Science and Engineering
University of New South Wales
Sydney, Australia 2052
mike@cse.unsw.edu.au

Abstract. In previous work it was shown that Formal Concept Analysis (FCA) can provide a useful framework for adjustment of representational bias for classifier learning and the construction of taxonomical hierarchies. This used techniques of predicate invention from Inductive Logic Programming (ILP) to introduce new attributes and re-formulate object descriptions. Such re-formulation of the descriptions of objects forces revision of the concept lattice. Hence a definition of revision operators based on ILP operators was introduced and shown to generate correct updates. However, in knowledge representation it is often the case that first-order or relational concepts are useful. Although there are previously published approaches to using first-order representations in FCA, in this paper we present an approach to constructing formal concepts using first-order logic intended to allow the application of methods developed in ILP. A lattice revision operator for relational concepts is then defined based on an ILP method for predicate invention.

1 Introduction

The descriptors used in Formal Concept Analysis (FCA) which form the intents of formal concepts are usually regarded as “attributes” of the “objects” which form the extents of formal concepts. The underlying binary relation, which together with the sets of objects and attributes denote the formal context, represents the fact that certain objects have a certain attributes. Many FCA applications have used descriptors for objects which are Boolean or many-valued attributes (in the latter case scaling is used). Sets of such descriptors can be thought of as conjunctions of variables in propositional logic.

Some problems have a naturally relational representation which is difficult to capture using propositional formalisms. For instance, say we have a chess end-game with only white king and rook and black king, which is known as the KRK end-game [10]. Applying symmetry, a simple pattern is that with black to move a position ($\mathbf{wkf}(\mathbf{Kf}), \mathbf{wkr}(3), \mathbf{wrf}(\mathbf{Rf}), \mathbf{wrr}(1), \mathbf{bkf}(\mathbf{Kf}), \mathbf{bkr}(1), \mathbf{diff}(\mathbf{Kf}, \mathbf{Rf}, \mathbf{D})$, $\mathbf{D} \neq 1$) is checkmate¹. Here we use Prolog notation with vari-

¹ Pieces given as (x, y) coordinates denoting file and rank, with \mathbf{wkf} , \mathbf{wkr} for the white king, \mathbf{wrf} , \mathbf{wrr} for the white rook and \mathbf{bkf} , \mathbf{bkr} for the black king; \mathbf{diff} is a relation giving the absolute distance between two files or ranks.

ables indicated by names beginning with an upper-case letter. The key part of the first-order or relational description in this example is that the two kings are on the same file, although the actual values for the file variables are not specified; additional constraints are supplied by the `diff` relation and the constants in the expression, for example indicating that the rook is attacking the black king. Taken together, the description refers to a concept denoting a set of objects, here chess positions, which can be concisely defined using relations between the properties of object components, here chess pieces. Patterns of similar form are not uncommon in real-world applications.

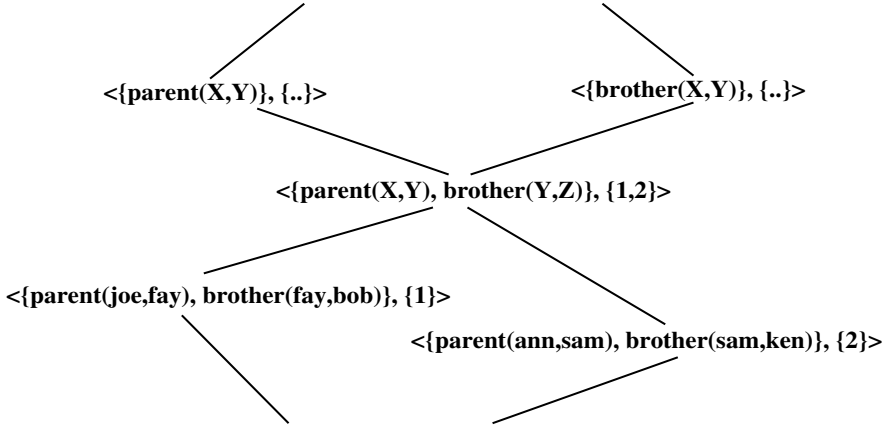
We are interested not only in the identification of interesting relational concepts but in using them as the basis for operators of predicate invention. Here we assume an underlying theory in a subset of the predicate calculus for which a formal context is constructed. In predicate invention the concept description is used in the definition of a new predicate not currently appearing in the theory, and hence not appearing in the description of any object in the formal context either. This forces the revision of the theory and the formal context, and thus the concept lattice. However, by doing so the theory becomes structured, redundancy is removed, and the new predicate is available for re-use. This approach was applied in a propositional logic framework to the problem of learning ontological rules [2]. In this paper we extend the approach to a first-order logic representation.

The outline of the paper is as follows. First, definitions are given for the framework of first-order logic formal concept analysis to be used in this work. Second, we define a revision operator on the concept lattice based on application of the standard ILP predicate invention operator inter-construction. We omit proofs of the correctness of this revision operator due to lack of space. Also not included are details of implementation and experimental results from its application. These are planned for a more comprehensive report as part of future work.

2 First-Order Logic Formal Concept Analysis

Several authors have investigated formal concept analysis in first order logic (see references in [5]). However, since every complete lattice is isomorphic to some concept lattice, it has been argued that the use of first-order logic as a vocabulary of descriptors does not extend FCA [6]. Nonetheless, a simple example illustrates that, for the same set of objects, using a first-order language for intents makes it straightforward to represent a concept which might be missed using a propositional formalism.

Example 1. We continue with the KRK chess end-game, using the same notation. The purpose of the example is to contrast the set of concepts generated under a propositional representation with those using a first-order representation. Suppose there are three chess positions which are the objects in our domain. These are:



- 1:** ... **parent(joe,fay), brother(fay,bob)** ...
2: ... **parent(ann,sam), brother(sam,ken)** ...

Fig. 1. A partial first-order concept lattice with relational intents, using Prolog syntax. Also shown in the lower part of the figure are objects 1 and 2 with partial descriptions.

(wkr(c),wkr(4),wrf(a),wrr(3),bkf(a),bkr(1))
(wkf(a),wkr(3),wrf(b),wrr(4),bkf(b),bkr(2))
(wkf(b),wkr(2),wrf(c),wrr(1),bkf(d),bkr(3))

Using a propositional representation where concept intents are formed from the set of attributes common to objects we have only the three concepts corresponding to these objects, apart from \top and \perp , in the lattice.

However, adopting a first-order logic representation where concept intents are the least general generalisations (or least common subsumers) of the descriptors common to objects we have in addition to the above the following concept:

(wkr(A),wkr(B),wrf(C),wrr(D),bkf(C),bkr(E))

Here the variable C is shared between the literals **wrf** and **bkf**, indicating that both pieces must have the same value in any positions covered by this concept (the first two in this case). Clearly, for the same set of objects, different representation languages may generate different sets of concepts.

In this paper, since we are interested in logic programming representations, our development is most related to those of [3,5]. Definitions are based, for FCA, on [7,4], for Logic Programming, on [9] and for ILP, on [11]. However, some naming and other conventions present in those sources have been changed to simplify the exposition.

Definition 1. First-order logic formal context. *A first-order logic formal context is a 4-tuple $\langle \mathcal{O}, \mathcal{D}, \mathcal{R}, \mathcal{Q} \rangle$. \mathcal{O} is a set of objects, \mathcal{D} is a set of descriptors, \mathcal{R} is a binary relation such that $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{D}$ and \mathcal{Q} is a meta-theory relating subsets of formulae from \mathcal{D} .*

It is assumed all objects can be uniquely identified. Descriptors are formalised in some subset of first-order predicate calculus. For example, we will assume in the remainder of this paper that descriptors are restricted to first-order function-free literals.

Definition 2. Description. *$D \subseteq \mathcal{D}$ is a description.*

In this paper we will assume that \mathcal{Q} defines a user-specified order. This could be founded on a variant of subsumption typically used in ILP [8]. We employ the following definition, based on subsumption of first-order formulae [12].

Definition 3. Subsumption. *Let $D_1, D_2 \subseteq \mathcal{D}$ be descriptions. D_1 subsumes D_2 , denoted $D_1 \succeq D_2$, if and only if there exists a substitution θ such that $D_1\theta \subseteq D_2$.*

Viewing a clause as a set of literals, the ordering of descriptions is isomorphic to the clause subsumption lattice [13]. The least general generalisation (lgg) C of two descriptions D_1 and D_2 is defined as the greatest lower bound in the lattice induced by \succeq [12].

Now we relate descriptions to objects, in the sense that an object can be said to have a description.

Definition 4. Object description. *Let $a \in \mathcal{O}$ be an object. Then the description of a is given by $\text{desc}(a) = \{b \in \mathcal{D} \mid \langle a, b \rangle \in \mathcal{R}\}$.*

For sets of objects, their descriptions are to be related via \mathcal{Q} . Here we use least general generalisation.

Definition 5. Generalised description. *For two objects a_1, a_2 with descriptions $\text{desc}(a_1), \text{desc}(a_2)$, respectively, the generalised description is given by $\text{gen}(a_1, a_2) = \text{lgg}(\text{desc}(a_1), \text{desc}(a_2))$. For n objects a_1, a_2, \dots, a_n the generalised description is given by $\text{gen}(a_1, a_2, \dots, a_n) = \text{lgg}(\text{desc}(a_1), \text{gen}(a_2, \dots, a_n))$.*

The set of generalised descriptions with respect to \mathcal{D} and \mathcal{Q} is $\mathcal{D}_{\mathcal{Q}}$.

Definition 6. First-order logic formal concept. *A first-order logic formal concept is an ordered pair of sets, written $\langle A, B \rangle$, where $A \subseteq \mathcal{O}$ and $B \in \mathcal{D}_{\mathcal{Q}}$. Each pair must be complete with respect to \mathcal{R} , which means that $A' = B$ and $B' = A$. Letting $A = \{a_1, a_2, \dots, a_n\}$, $A' = \text{gen}(a_1, a_2, \dots, a_n)$. $B' = \{a \in \mathcal{O} \mid B \succeq \text{desc}(a)\}$.*

From now on we will omit the prefix “first-order logic” for formal contexts and formal concepts except to resolve ambiguity.

Although the introduction of a meta-theory \mathcal{Q} as a parameter in the formalisation may seem unusual, experience in ILP has shown the importance of

allowing flexibility in dealing with ordering relationships involving formulae in first-order logic. Essentially this is due to the inherent complexity of first-order logic languages. In order to build practical systems, therefore, it is usually necessary to apply both syntactic and semantic constraints to the form of expressions allowed. This can be done in a transparent way using so-called “declarative bias”, essentially in a meta-theory. Thus our addition of \mathcal{Q} is an important but conservative extension to FCA which will enable practical ILP applications such as [14].

Example 2. To see one reason why the intermediate step of generating all descriptions for objects, then taking their lgg, is used to define the maps $'$, imagine we had the case where a concept has intent $B = \{p(X, Y), q(Y, Z)\}$. Suppose the object 1 had only the following elements in \mathcal{R} : $\langle 1, p(a, b) \rangle$ and $\langle 1, q(c, d) \rangle$. Then we should not want 1 to appear in the extent of the concept with intent B . But selecting each literal separately from B on the basis of a set construction $\forall b \in B$ would allow 1 to be in the extent. This is because there is no connection between the substitution for variable Y in the first literal, Y/b , and the substitution for variable Y in the second, Y/c . By considering a description, i.e., set of descriptors, we can collect all objects where the second argument of descriptor p contains the same term as the first argument of descriptor q . Then $B \not\leq \text{desc}(1)$. However, if instead \mathcal{R} had only $\langle 1, p(a, b) \rangle$ and $\langle 1, q(b, c) \rangle$ for object 1 then 1 should appear in the corresponding extent using the same method, which is correct.

The correspondence between intent and extent of complete concepts is a Galois connection between the power set $2^{\mathcal{O}}$ of the set of objects and the set $\mathcal{D}_{\mathcal{Q}}$ of generalised descriptions. The Galois lattice \mathcal{L} for the binary relation is the set of all complete pairs of intents and extents, with the following partial order. Given two concepts $N_1 = \langle A_1, B_1 \rangle$ and $N_2 = \langle A_2, B_2 \rangle$, $N_1 \leq N_2 \leftrightarrow A_1 \subseteq A_2$. The dual nature of the Galois connection means we have the equivalent relationship $N_1 \leq N_2 \leftrightarrow B_2 \vdash_{\mathcal{Q}} B_1$. The set of formal concepts of the formal context $\langle \mathcal{O}, \mathcal{D}, \mathcal{R}, \mathcal{Q} \rangle$ together with \leq is a complete lattice. From Definition 5, this lattice is isomorphic to the clause subsumption lattice. An example of a notional partial lattice is shown in Figure 1.

3 Revision of \mathcal{L} by Inter-construction

The first order logic version of the propositional logic inverse resolution operator used in [1] can now be developed. A generalised inter-construction operator can be defined either in terms of the least general generalisation of the inverse linear derivations of pairs of clauses or as the relative least general generalisation of pairs of clauses [11]. Taking the latter approach, we can define updates to the first-order concept lattice similarly to those defined for the propositional version. The operator is: $C^+ = \text{lgg}(D_i, D_j) \cup \{l\}$ where D_i, D_j are the given implicates, C^+ is the invented implicant, and l is a literal, an instance of the “invented” predicate.

Applying inter-construction to a set of object descriptions, as sketched in Figures 1 and 2, when treated as a theory, does not have any generalising or specialising effect. This is because as a predicate-invention operator, it is truth-preserving. However, it does change the set of generalised descriptions. Consequently, the concept lattice must be revised to reflect this.

The lattice is in state $t - 1$ preceding revision and state t following revision. The concepts (henceforth nodes) in the concept lattice to be revised may be partitioned into different types with respect to the goal node, which is the concept selected as the basis for revision by inter-construction. Types are denoted as follows: goal g , ancestor a , descendant d . These node types are related by subsumption. Those not related by subsumption but with some partial overlap are either related by intent and extent, rIE , related by intent only, rI , or related by extent only, rE . A single node has the type of the newly invented predicate, i . The intent of a node at $t - 1$ is denoted $I(n_{t-1})$ and the extent of a node at t is denoted $E(n_t)$, where n is a node type.

Definition 7. Revising \mathcal{L}_{t-1} with inter-construction

$$\begin{aligned} g_t &= \langle I(g_{t-1}), \{k\} \rangle \\ a_t &= \langle I(a_{t-1}) \cup X, E(a_{t-1}) \setminus E(g_{t-1}) \rangle \\ d_t &= \langle I(d_{t-1}) \setminus I(g_{t-1}) \cup \{l\}, E(d_{t-1}) \rangle \\ rI_t &= \langle I(rIE_{t-1}) \cup X, E(rIE_{t-1}) \setminus E(g_{t-1}) \rangle \\ rE_t &= \langle I(rIE_{t-1}) \setminus I(g_{t-1}), E(rIE_{t-1}) \cup Y \rangle \\ i_t &= \langle \{l\}, E(g_{t-1}) \rangle \end{aligned}$$

In this definition k is the index of a new object having a description subsumed by C^+ , and X and Y are required to complete the intent and extents, respectively, of revised formal concepts. An example of the concept lattice shown in Figure 1 which has been revised following predicate invention (the new relation “uncle” defined in terms of “parent” and “brother”) is in Figure 2.

3.1 Related Work

In Chaudron and Maille [3] intents of formal concepts are “cubes”, i.e. existentially-quantified conjunctions of first-order literals. The lattice of cubes, where the infimum and supremum are defined using logical reduction (based on theta subsumption) and anti-unification (to compute least general generalisations), leads to a definition of first-order FCA. Under this framework unique least general generalisations exist but may (in the worst case) have infinite cardinality, even after reduction [13]. The cardinality of least general generalisations relative to background knowledge can be exponential in the number of instances.

Instead, we have adopted the approach of constraining the derivability relation between expressions defined by a theory at the meta-level. One successful example of this is reported by Sammut [14], in which Prolog is used to efficiently compute syntactic and semantic constraints on induced expressions, thus greatly simplifying the output generalisations.

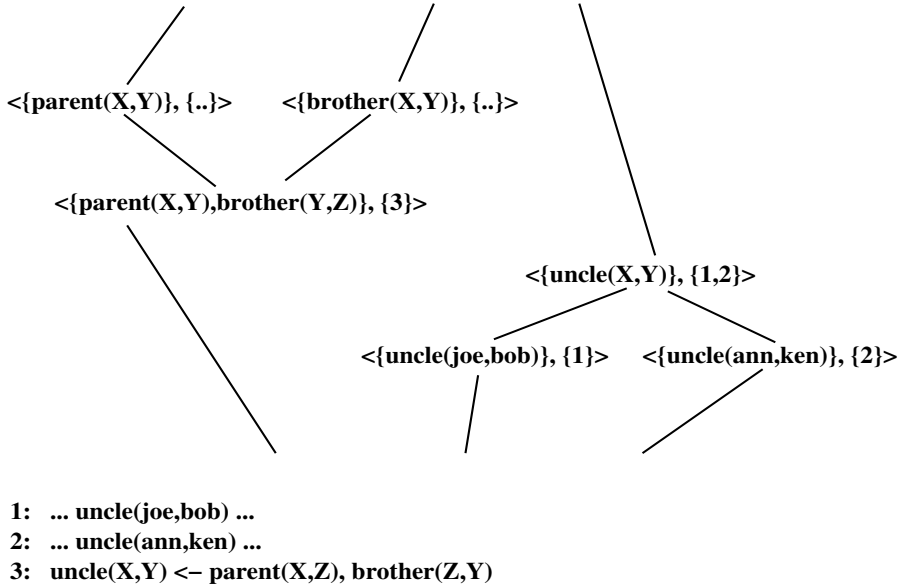


Fig. 2. The revised partial lattice from Fig. 1 including the invented predicate “uncle” which has a new definition in the description for object 3.

A more general approach to combining logic and FCA has been developed by Ferré and Ridoux [5]. They reformulate FCA as Logical Concept Analysis. Instead of adding an extra parameter to formal contexts defining derivability relationships between intents formulated as logical expressions as we do, they add an entire logic with ordering relationships to replace attributes in formal contexts. These logical contexts lead to a formalisation called contextualized logic. The relationship between this formalism and ours will be further investigated.

Ganter and Kuznetsov [6] have presented arguments that the above-cited approaches cannot be said to generalize FCA, since the basic framework is sufficiently general already, in the sense that every complete lattice is isomorphic to some concept lattice. However, they argue that such approaches may be warranted on the grounds of efficiency. Moreover we have shown in Example 1 that, for the same set of objects, a first-order formalism may enable the representation of concepts not generated when intent descriptors are restricted to propositional variables. The relation between the patterns in pattern structures [6] and logical expressions as used in this paper will also be further investigated.

4 Conclusions and Further Work

Although this work is preliminary, we have presented the basic framework to lift our previous work on predicate invention [2] in FCA to a first-order logic

representation. This will open up a wide area of techniques from ILP to be investigated within FCA. A prototype implementation is being developed and so far works as expected on the types of examples presented. Further experimentation is required to test the assumption that techniques from ILP can be developed to make the approach practical. This work already has provoked questions for further theoretical investigation, however, such as the apparent asymmetry in the status of descriptions and objects in first-order formal concept analysis; the former have a structural ordering where the latter do not.

References

1. M. Bain. Structured Features from Concept Lattices for Unsupervised Learning and Classification. In B. McKay and J. Slaney, editors, *AI 2002: Proc. of the 15th Australian Joint Conference on Artificial Intelligence*, LNAI 2557, pages 557–568, Berlin, 2002. Springer.
2. M. Bain. Learning Ontologies from Concept Lattices. In B. Ganter and A. de Moor, editors, *Using Conceptual Structures: Contributions to ICCS 2003*, pages 199–212, Aachen, Germany, 2003. Shaker Verlag.
3. L. Chaudron and N. Maille. 1st Order Logic Formal Concept Analysis: from logic programming to theory. *Linkoping Electronic Articles in Computer and Information Science*, 3(13), 1998.
4. B.A. Davey and H.A. Priestley. *An Introduction to Lattices and Order (Second Edition)*. Cambridge University Press, 2002.
5. S. Ferré and O. Ridoux. A Logical Generalization of Formal Concept Analysis. In Guy Mineau and Bernhard Ganter, editors, *Proc. Eighth Intl. Conf. on Conceptual Structures*, Berlin, 2000. Springer.
6. B. Ganter and S. Kuznetsov. Pattern Structures and their Projections. In H. Delugach and G. Stumme, editors, *Proceedings of the International Conference on Conceptual Structures, LNCS 2120*, pages 129–142, Berlin, 2001. Springer.
7. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin, 1999.
8. J-U. Kietz and M. Lübke. An Efficient Subsumption Algorithm for Inductive Logic Programming. In S. Wrobel, editor, *Proc. Fourth Intl. Workshop on Inductive Logic Programming*, pages 97–106. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.
9. J. W. Lloyd. *Logic Programming, 2nd Edition*. Springer-Verlag, Berlin, 1987.
10. D. Michie. King and Rook Against King: Historical Background and a Problem on the Infinite Board. In M. R. B. Clarke, editor, *Advances in Computer Chess*, volume 1, pages 30–59. Edinburgh University Press, Edinburgh, 1977.
11. S. Muggleton. Inverse Entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
12. S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, *Inductive Logic Programming*, pages 281–298. Academic Press, London, 1992.
13. G. D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, August 1971.
14. C. Sammut. Prolog, Refinements and RLGG. In D. Page, editor, *Proc. Eighth Intl. Conference on Inductive Logic Programming*, pages 225–234, Berlin, 1998. Springer.

The Power of Peircean Algebraic Logic (PAL)

Joachim Hereth Correia¹ and Reinhard Pöschel²

¹ Darmstadt University of Technology
Department of Mathematics, Schlossgartenstr. 7
D-64289 Darmstadt, Germany
`hereth@mathematik.tu-darmstadt.de`

² Dresden University of Technology
Department of Mathematics, Institute for Algebra
D-01062 Dresden, Germany
`poeschel@math.tu-dresden.de`

Abstract. The existential graphs devised by Charles S. Peirce can be understood as an approach to represent and to work with relational structures long before the manifestation of relational algebras as known today in modern mathematics. Robert Burch proposed in [Bur91] an algebraization of the existential graphs and called it the *Peircean Algebraic Logic* (PAL). In this paper, we show that the expressive power of PAL is equivalent to the expressive power of Krasner-algebras (which extend relational algebras). Therefore, from the mathematical point of view these graphs can be considered as a two-dimensional representation language for first-order formulas. Furthermore, we investigate the special properties of the teridentity in this framework and Peirce's thesis, that to build all relations out of smaller ones we need at least a relation of arity three (for instance, the teridentity itself).

Introduction

For C.S. Peirce, the main purpose of logic as a mathematical discipline was to analyze and display the fundamental constituents of reasoning in an easily understandable fashion. For this, he preferred algebraic logic to the quantificational logic whose development started at his time (and has even been partly influenced by him). Today, mathematical logic is often understood as quantificational logic. Peirce however developed a graphical language to express logic, the *existential graphs* which in his sense are better suited for the task of logic than the formulas of quantificational logic. In this paper, we elaborate that existential graphs can be considered as two-dimensional representations of logical statements.

In [Bur91] Robert Burch proposes a formulation of Peirce's existential graphs adapted to modern algebra. He calls this formalization Peircean Algebraic Logic (PAL) and proposes a graphical representation of the terms of PAL which resemble Peirce's existential graphs. His work also inspired the development of the *Contextual Logic of Relations* (see [Wil00], [Arn01], [Pol02]). For this paper, we

consider a modified version of PAL. We adopt Burch's analysis of the fundamental operations of PAL, but while Burch allows juxtaposition (which corresponds to the cross product) only at the end of the process of term construction, we allow it at arbitrary places. Due to this modification, the proofs Burch is providing for Peirce's thesis do not hold anymore. Also, we will adapt a more traditional (in the mathematical sense) approach using only the concept of the "evaluation" (or "interpretation") of a term instead of the concepts "depiction", "representation", "expression" and "denotation" as used by Burch. We thereby concentrate on the mathematical properties of PAL, deliberately ignoring the philosophical argumentation by Burch.

Organization of This Paper

In the following section we will briefly introduce some basic definitions and results from relational algebra. In Section 2 we present an analogous summary of the Peircean Algebraic Logic and its graphical representation as well as the main theorem on the equivalence of expressive power. Section 3 treats the special properties of the teridentity and the necessity of ternary relations for the construction of all relations (Peirce's thesis).

Then we consider an alternative system for the generation of PAL and conclude the paper with some remarks on the influence of PAL on the Contextual Logic of Relations and further work in this area.

1 Relational Algebras

In this section, we briefly recall the basic definitions for relational and Krasner algebras and their equivalence to systems of first-order logic formulas.

Notations 1 (Operations and Relations).

We introduce for operations $f : A^n \rightarrow A$ and relations $\varrho \subseteq A^m$ on a fixed base set A with $m, n \in \mathbb{N}_0$ and $m \geq 0, n \geq 1$ the following notation:

$$\begin{aligned}
 \text{Op}^{(n)}(A) &:= \{f \mid f : A^n \rightarrow A\} && (n\text{-ary operations}) \\
 \text{Op}(A) &:= \bigcup_{n=1}^{\infty} \text{Op}^{(n)}(A) && (\text{finitary operations}) \\
 \text{Sym}(A) &:= \{f \in \text{Op}^{(1)}(A) \mid f \text{ bijectiv}\} && (\text{permutations}) \\
 \text{Rel}^{(m)}(A) &:= \{\varrho \mid \varrho \subseteq A^m\} && (m\text{-ary relations}) \\
 \text{Rel}(A) &:= \bigcup_{m=0}^{\infty} \text{Rel}^{(m)}(A) && (\text{finitary relations}) \\
 \mathfrak{P}(A) &:= \text{Rel}^{(1)}(A) && (\text{power set})
 \end{aligned}$$

The arity of an operation f or relation ϱ will be denoted by $\text{ar}(f)$ or $\text{ar}(\varrho)$, resp. In this article, 0-ary functions are not considered mainly for technical reasons (which appear in connection with clones of operations but play no role in this paper). Note that the elements of $\text{Rel}^{(0)}(A)$ (0-ary relations) are the subsets of $A^0 = \{\emptyset\}$, i. e. the two relations \emptyset and $\{\emptyset\}$.

An m -tuple $r \in A^m$ sometimes will be regarded as a mapping $r : \underline{m} \rightarrow A$ with $\underline{m} := \{1, \dots, m\}$, and its components are given by $r = (r(1), \dots, r(m))$.

Definitions 2 (Relational algebras and clones).

We consider the following (set-theoretical) operations on relations (ϱ and σ denote arbitrary m -ary and s -ary relations on A , resp.):

(R1) *Diagonal relation* Δ_A (nullary operation: to contain the equality relation $\Delta_A := \{(a, a) \mid a \in A\}$),

(R2a) *Cyclic shift of coordinates* ζ :

$$\zeta\varrho := \{(a_1, \dots, a_m) \mid (a_2, \dots, a_m, a_1) \in \varrho\},$$

(R2b) *Transposition of first two coordinates* τ :

$$\tau\varrho := \{(a_1, \dots, a_m) \mid (a_2, a_1, a_3, \dots, a_m) \in \varrho\},$$

(R3) *Identification of the first two coordinates* Δ :

$$\Delta\varrho := \{(a_1, \dots, a_{m-1}) \mid (a_1, a_1, a_2, \dots, a_{m-1}) \in \varrho\},$$

(R4) *Relational product* \circ :

$$\varrho \circ \sigma := \{(a_1, \dots, a_{m+s-2}) \mid \exists b \in A : (a_1, \dots, a_{m-1}, b) \in \varrho \text{ and } (b, a_m, \dots, a_{m+s-2}) \in \sigma\},$$

(R5) *Adding a fictitious last component* ∇ :

$$\nabla\varrho := \{(a_1, \dots, a_m, b) \in A^{m+1} \mid (a_1, \dots, a_m) \in \varrho \text{ and } b \in A\}.$$

(R6) *Union* (of relations of the same arity $m = s$): $\varrho \cup \sigma$,

(R7) *Complementation*: $\neg\varrho := A^m \setminus \varrho$

A set $Q \subseteq \text{Rel}(A)$ of relations is called *relational algebra*, *weak Krasner algebra* or *Krasner algebra*, respectively, if Q is closed with respect to (R1)–(R5), (R1)–(R6) or (R1)–(R7), respectively (these definitions were used e. g. in [PösK79, 1.1.8]). The corresponding closures will be denoted by $\langle Q \rangle_{\text{RA}}$, $\langle Q \rangle_{\text{WKA}}$ and $\langle Q \rangle_{\text{KA}}$, resp. For finite sets A these algebras are also called *clones* (relational clone, (weak) Krasner clone).

Please note that relational algebras contain relations of arbitrary (finite) arity and are much more general than Tarski's relation algebras as introduced in [Tar41].

Definitions 3 (Further operations on relations). There are many operations which can be derived from (R1)–(R7) given above (see also Def. 5 below), we mention some of them (again, ϱ and σ have arity m and s , resp.):

- (R8) *Diagonal relations:* Let ε be a partition of $\{1, \dots, m\}$. Then the m -ary relation ($m \in \mathbb{N}_+$)

$$d_\varepsilon^A := \{(a_1, \dots, a_m) \in A^m \mid i \equiv_\varepsilon j \implies a_i = a_j\}$$

is called *diagonal* relation (where $i \equiv_\varepsilon j$ means that i and j belong to the same block of ε). Let D_A denote the set of all diagonal relations on A .

Special diagonal relations are the binary equality relation $\Delta_A = d_{\{1,2\}}$ and the ternary teridentity $\text{id}_3^A = d_{\{1,2,3\}}$ (which plays a central role in PAL, see Section 2).

- (R9) *Permutation of coordinates:* For a permutation α on the set $\{1, \dots, m\}$ let

$$\pi_\alpha(\varrho) := \{(a_1, \dots, a_m) \mid (a_{\alpha(1)}, \dots, a_{\alpha(m)}) \in \varrho\}.$$

- (R10) *Projection* onto a subset I of coordinates: For $I = \{i_1, \dots, i_t\}$ with $1 \leq i_1 < i_2 < \dots < i_t \leq m$ we define

$$\text{pr}_I(\varrho) := \{(a_{i_1}, \dots, a_{i_t}) \mid \exists a_j (j \in \{1, \dots, m\} \setminus I) : (a_1, \dots, a_m) \in \varrho\}.$$

- (R11) *Coupled deletion of coordinates:* For $1 \leq i < j \leq m$ let

$$\begin{aligned} \delta_{ij}(\varrho) := \{ & (a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_{j-1}, a_{j+1}, \dots, a_m) \in A^{m-2} \mid \\ & \exists b \in A : (a_1, \dots, a_{i-1}, b, a_{i+1}, \dots, a_{j-1}, b, a_{j+1}, \dots, a_m) \in \varrho \}. \end{aligned}$$

In case of $m = 2$ this definition is evaluated as follows:

$$\delta_{ij}(\varrho) := \begin{cases} \{\emptyset\} & \text{if } \exists b \in A : (b, b) \in \varrho, \\ \emptyset & \text{otherwise.} \end{cases}$$

- (R12) *Intersection* of relations of the same arity $m = s$: $\varrho \cap \sigma$.

- (R13) *Product* (cartesian or cross product):

$$\begin{aligned} \varrho \times \sigma := \{ & (a_1, \dots, a_m, b_1, \dots, b_s) \in A^{m+s} \mid (a_1, \dots, a_m) \in \varrho, \\ & (b_1, \dots, b_s) \in \sigma \}. \end{aligned}$$

Remark 4. It is straightforward to show that all these operations (R8)–(R13) can be derived from the operations Def. 2 (R1)–(R5) (for instance, a cyclic shift and a transposition generate all permutations, thus ζ and τ generate all π_α ; $\varrho \times \sigma = (\nabla \varrho) \circ (\zeta \nabla \sigma)$ and so on, for more details see [PösK79, 1.1.9]). The interconnections of the operations (R1)–(R13) are discussed further in Section 4.

Definitions 5 (Logical operations and closures).

Relational algebras can be characterized also via closure with respect to first order formulas. To each first order formula $\varphi(R_1, \dots, R_q; x_1, \dots, x_m)$ with free

variables x_1, \dots, x_m and m_i -ary relation (predicate) symbols R_i ($i \in \{1, \dots, q\}$) one can assign an operation (called *logical operation*)

$$L_\varphi : \text{Rel}^{(m_1)}(A) \times \dots \times \text{Rel}^{(m_q)}(A) \rightarrow \text{Rel}^{(m)}(A) \quad \text{with}$$

$$L_\varphi(\varrho_1, \dots, \varrho_q) := \{(a_1, \dots, a_m) \in A^m \mid \models \varphi(\varrho_1, \dots, \varrho_q; a_1, \dots, a_m)\}.$$

Hereby, for atomic formulas e.g. $R_i(x, y)$ we interpret $\models \varrho_i(a, b)$ as $(a, b) \in \varrho_i$ (for elements $a, b \in A$). For instance, the formula

$$\varphi(R_1, R_2; x_1, x_2) := \exists z : R_1(x_1, z) \wedge R_2(z, x_2)$$

(with binary relation symbols) induces the operation relational product (see Def. 2 (R4)):

$$L_\varphi(\varrho_1, \varrho_2) = \{(a_1, a_2) \mid \exists z \in A : (a_1, z) \in \varrho_1 \wedge (z, a_2) \in \varrho_2\} = \varrho_1 \circ \varrho_2$$

for binary relations $\varrho_1, \varrho_2 \in \text{Rel}^{(2)}(A)$.

Let $\Phi(\exists, \wedge, \dots)$ denote the set of all first order formulas that contain only the indicated quantifier \exists , the indicated connectives \wedge, \dots and relation symbols and variables. Let $\text{Lop}_A(\exists, \wedge, \dots) := \{L_\varphi \mid \varphi \in \Phi(\exists, \wedge, \dots)\}$ denote the corresponding logical operations.

Then we have (see [PösK79, 2.1.3]):

Theorem 6. *Let $Q \subseteq \text{Rel}(A)$. Then*

$$\begin{aligned} Q \text{ relational algebra} &\iff Q \text{ closed w.r.t. } \text{Lop}_A(\exists, \wedge, =), \\ Q \text{ weak Krasner algebra} &\iff Q \text{ closed w.r.t. } \text{Lop}_A(\exists, \wedge, \vee, =), \\ Q \text{ Krasner algebra} &\iff Q \text{ closed w.r.t. } \text{Lop}_A(\exists, \wedge, \vee, \neg, =). \end{aligned}$$

2 Peircean Algebraic Logic (PAL)

The operations of the *Peircean Algebraic Logic and relation graphs* (PAL) are closely related to the existential graphs that Peirce developed in the late 1890s. Here we shall not go into details of existential graphs (which are also related to the conceptual graphs, see [Sow92]) and only describe the Peircean operations interpreted as operations on relations and show the corresponding graphs representing such operations (these operations on relations were also proposed by R. W. Burch in [Bur91] and modelled mathematically in power context families, see [Wil00], [Arn01] and [Pol02]).

Definitions 7 (Relation Graphs and Operations of PAL).

An m -ary relation $\varrho^A \in \text{Rel}^{(m)}(A)$ will be represented graphically by a point (dot or small circle) with m outgoing “arms”, i. e. pending edges (called *hooks* in [Bur91]), where each pending edge has exactly one label from $\{1, \dots, m\}$, see Fig. 1 (a); the dot itself is labeled by ϱ . All these labels are called *relation labels*

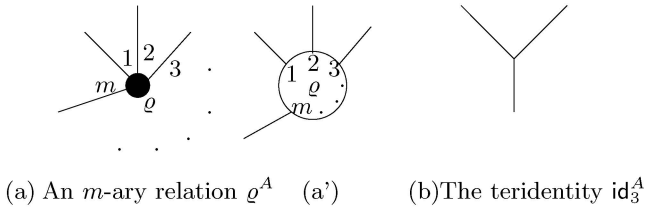


Fig. 1. Graphical representation of relations in PAL

(they belong to the relation symbol ϱ and indicate to which “spot” (=component) of ϱ an incoming edge is connected, Fig. 1(a') makes this more clear; but we prefer the less complicated drawing of (a)). The order of the labels of the edges is not essential: two figures with the same labels in possibly different positions in the picture represent the same relation.

The so-called *teridentity* is the ternary relation

$$\text{id}_3 := \text{id}_3^A := \{(a, a, a) \mid a \in A\}$$

and will be represented as in Fig. 1 (b) (no relation label is necessary).

Let us fix some convention: the figures to be constructed shall be called *relation graphs* ([Pol02]). They have some “outgoing” pending edges, its number is called the *arity* of the relation graph. The pending edges of an m -ary relation graph are labelled by the elements $1, \dots, m$ (we shall call them *graph labels* in order to distinct them from the relation labels).

The constructions of PAL can be described on purely syntactical level by providing a set Σ (signature) of relation symbols (with arities) for labeling the points. However, if there is an *evaluation* (interpretation) of each relation symbol by a concrete relation (of the same arity) on a fixed base set A (i. e. we have an arity-preserving *evaluation function* $\Sigma \rightarrow \text{Rel}(A) : \varrho \mapsto \varrho^A$), then to each m -ary relation graph \mathfrak{G} uniquely corresponds an m -ary relation \mathfrak{G}^A on A .

In the following we describe constructions (allowed by PAL) for relation graphs in more detail. We shall give the construction which at the same time provides a *constructive definition of relation graphs* \mathfrak{G} and the evaluation as relation (denoted by \mathfrak{G}^A). However we do this by representing relation graphs by figures. From the pure mathematical point of view, a relation graph is a multi-graph with ordered valencies, pending edges and nested subgraphs as defined in [Pol02, Def. 2-8] (also called *structure* there), we try to avoid such technicalities and refer to [Pol02].

- (PAL1) atomic relation graphs: Let $\varrho \in \Sigma$ be an m -ary relation symbol. Then the graph (also denoted by ϱ) in Fig. 2 (a) is an m -ary relation graph. It has m pending edges each of which has a relation label (drawn near the center) and a graph label (drawn at the outer end) which coincides with the relation label. Its corresponding relation (evaluation on base set A) is ϱ^A . The graph in Fig. 2 (b) is the relation graph of the teridentity. It has the graph labels 1, 2, 3 for its pending edges. The evaluation is the relation id_3^A .

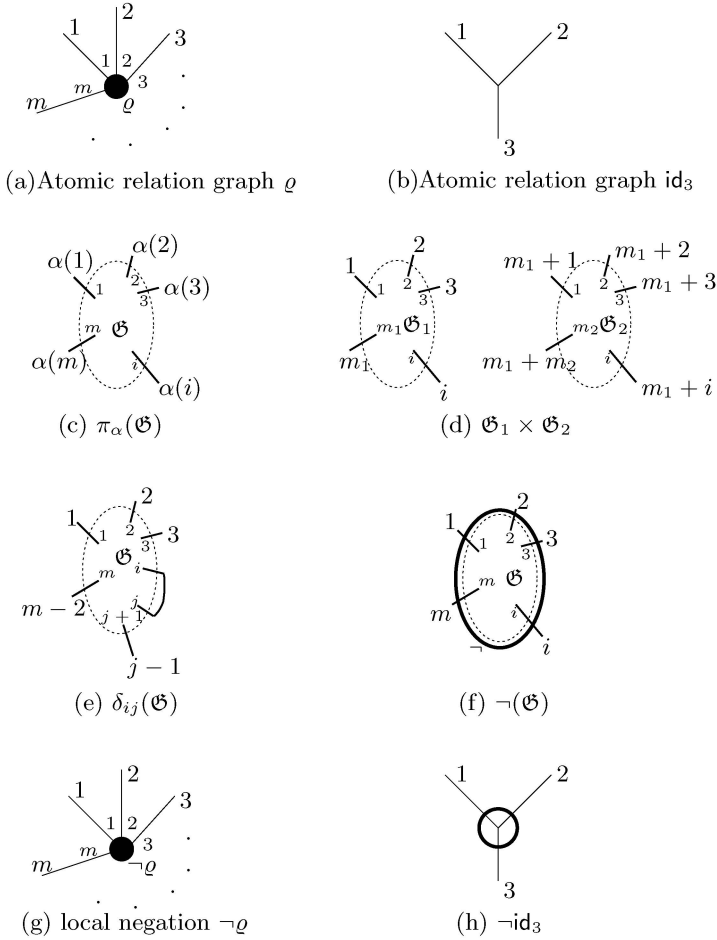


Fig. 2. Construction of relation graphs. The small labels (drawn inside the dotted line) in (c)–(f) are the original graph labels of \mathfrak{G} and have to be replaced by the new graph labels drawn outside the dotted line

(PAL2) Permutation of pending edges: Let \mathfrak{G} be an m -ary relation graph with corresponding relation \mathfrak{G}^A and let $\alpha : \underline{m} \rightarrow \underline{m}$ be a permutation. Then the graph in Fig. 2 (c) obtained from \mathfrak{G} by changing the label i to $\alpha(i)$ for every graph label $i \in \underline{m} := \{1, \dots, m\}$ of a pending edge (outgoing half-edge), is an m -ary relation graph denoted by $\pi_\alpha(\mathfrak{G})$. Its evaluation is the relation

$$\pi_\alpha(\mathfrak{G})^A := \pi_\alpha(\mathfrak{G}^A) \text{ (see Def. 3 (R9)).}$$

(PAL3) Juxtaposition (cross product): Let \mathfrak{G}_i be m_i -ary relation graphs ($i = 1, 2$). Then the graph in Fig. 2 (d) obtained by juxtaposition and re-labeling the graph labels of the pending edges of \mathfrak{G}_2 (according to

$i \mapsto m_1 + i$ for $i \in \{1, \dots, m_2\}$) is an $(m_1 + m_2)$ -ary relation graph denoted by $\mathfrak{G}_1 \times \mathfrak{G}_2$. The evaluation is given by

$$(\mathfrak{G}_1 \times \mathfrak{G}_2)^A := \mathfrak{G}_1^A \times \mathfrak{G}_2^A \text{ (see Def. 3 (R13)).}$$

- (PAL4) Connecting pending edges: Let \mathfrak{G} be an m -ary relation graph with $m \geq 2$. Let $1 \leq i < j \leq m$. Then the graph in Fig. 2(e) obtained by connecting the pending edges with graph label i and j (deleting these graph labels and relabeling the other pending edges) is an $(m-2)$ -ary relation graph denoted by $\delta_{ij}(\mathfrak{G})$. The evaluation is given by

$$(\delta_{ij}(\mathfrak{G}))^A := \delta_{ij}(\mathfrak{G}^A) \text{ (see Def. 3 (R11)).}$$

- (PAL5) negation (complementation): Let \mathfrak{G} be an m -ary relation graph. Then the graph in Fig. 2(f) obtained from \mathfrak{G} by drawing a simple closed curve (called also negation circle, optionally labeled by \neg) enclosing the whole graph and prolonging the pending edges outside the curve (while keeping the labels) is a relation graph denoted by $\neg(\mathfrak{G})$. Its evaluation is

$$(\neg(\mathfrak{G}))^A := \neg(\mathfrak{G}^A) \text{ (see Def. 2 (R7)).}$$

- (PAL5') local negation: For an atomic relation graph ϱ where $\varrho \in \Sigma$, the relation graph $\neg(\varrho)$ is called locally negated atomic relation graph; it will be drawn simply also by changing the label from ϱ to $\neg\varrho$ (and not drawing the curve according to (PAL5)).

Remark: Note that local negation does not include the negation of the teridentity (which we shall represent as in Fig. 2(h) without relation labels). This is because the negation of the teridentity allows to simulate the (unrestricted) negation, i. e., (PAL5') would be “semantically” as powerful as (PAL5) (see Proposition 11 and Remark 12).

A *relation graph* is a graph obtained from the atomic relation graphs by using (PAL1)–(PAL5) finitely often. By construction, a relation graph has graph labels for its pending edges while each other edge has a relation label on each half that is connected with an atomic relation $\varrho \in \Sigma$ (the teridentity just “splits” one edge into three “directions” and needs no relation labels).

The set of all relation graphs over a signature Σ is denoted by $\langle \Sigma \rangle_{\text{PAL}}$.

The closure with respect to the rules (PAL1)–(PAL4) and (PAL5') (i. e. only local negation is allowed instead of arbitrary negation) will be denoted by $\langle \Sigma \rangle_{\text{InegPAL}}$ (“locally negated PAL”-closure). Finally, the closure with respect to (PAL1)–(PAL4) (without any negation) will be denoted by $\langle \Sigma \rangle_{\text{pPAL}}$ (“positive PAL”-closure). The relation graphs in $\langle \Sigma \rangle_{\text{InegPAL}}$ and $\langle \Sigma \rangle_{\text{pPAL}}$ are called *locally negated* and *positive* relation graphs, respectively.

Now the following operators on $\text{Rel}(A)$ can be introduced:

Let $Q \subseteq \text{Rel}(A)$ and let $\Sigma_Q := \{\underline{\varrho} \mid \varrho \in Q\}$ be the relational signature of abstract relation symbols such that $\underline{\varrho}$ has the same arity as ϱ , and the evaluation of $\underline{\varrho}$ over A is canonically given by $\underline{\varrho}^A := \varrho$. Then we define:

$$\langle Q \rangle_{\text{PAL}} := \{\mathfrak{G}^A \mid \mathfrak{G} \in \langle \Sigma_Q \rangle_{\text{PAL}}\}.$$

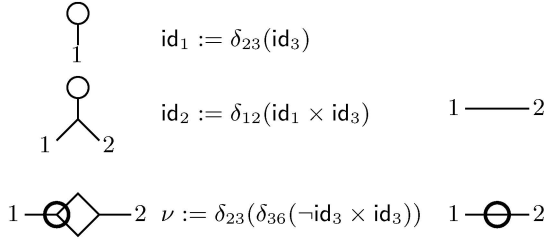


Fig. 3. Some relation graphs derivable from id_3

Analogously we define $\langle Q \rangle_{\text{InegPAL}}$ and $\langle Q \rangle_{\text{pPAL}}$. If there is danger of confusion the base set A is added as upper index: $\langle Q \rangle_{\text{PAL}}^A$.

Examples 8. Some examples of relation graphs composed by the above rules (PAL1)–(PAL5') are given in Figures 3 and 4, together with a term (in the language of relation graphs, see (PAL1)–(PAL5)) describing it.

In Fig. 3 we specify the relation graphs id_1 , id_2 and ν which are all generated by the teridentity id_3 (the describing term is always given) and introduce a more convenient graphical representation for them.

The evaluation of these relations graphs is obvious:

$$\text{id}_1^A = A \in \text{Rel}^{(1)}(A), \quad \text{id}_2^A = \Delta_A, \quad \nu^A = \{(a, b) \in A^2 \mid a \neq b\}.$$

In Fig. 4 we want to attract the attention to the generating process of relation graphs (the reader should check every step). While the graphical construction is immediate, it is easy but tedious to specify the corresponding describing terms (one carefully has to handle the labels). This demonstrates the advantage of the graphical presentation of relation graphs which is much closer to human thinking and understanding than the “linear” technical composition of terms. For instance, it immediately follows from the given drawing of the relation graphs in this figure that for a representation on any set A we have $\mathfrak{G}_6^A = \varrho^A \cap \sigma^A$ and $\mathfrak{G}_8^A = \neg(\neg\varrho^A \cap \neg\sigma^A) = \varrho^A \cup \sigma^A$. Note that there are many ways to construct other relation graphs which also represent uniformly (i. e. for all A) intersection and union of relations. e. g., one could take first the cross product $\text{id}_3 \times \varrho \times \sigma \times \text{id}_3$ (with 10 pending edges) and then do the necessary connections between pending edges to get \mathfrak{G}_6 .

The following theorem shows that the operations of relational algebras and the operations of PAL are equally powerful: for a given set Q of relations one can generate the same set of derived relations. According to Def. 5 there also exists an equivalent description by first order formulas.

Theorem 9. *Let $Q \subseteq \text{Rel}(A)$ and let $\neg Q := \{\neg\varrho \mid \varrho \in Q\}$. Then:*

- (i) $\langle Q \rangle_{\text{pPAL}} = \langle Q \rangle_{\text{RA}}$,
- (ii) $\langle Q \rangle_{\text{InegPAL}} = \langle Q \cup \neg Q \rangle_{\text{RA}}$,
- (iii) $\langle Q \rangle_{\text{PAL}} = \langle Q \rangle_{\text{KA}}$.

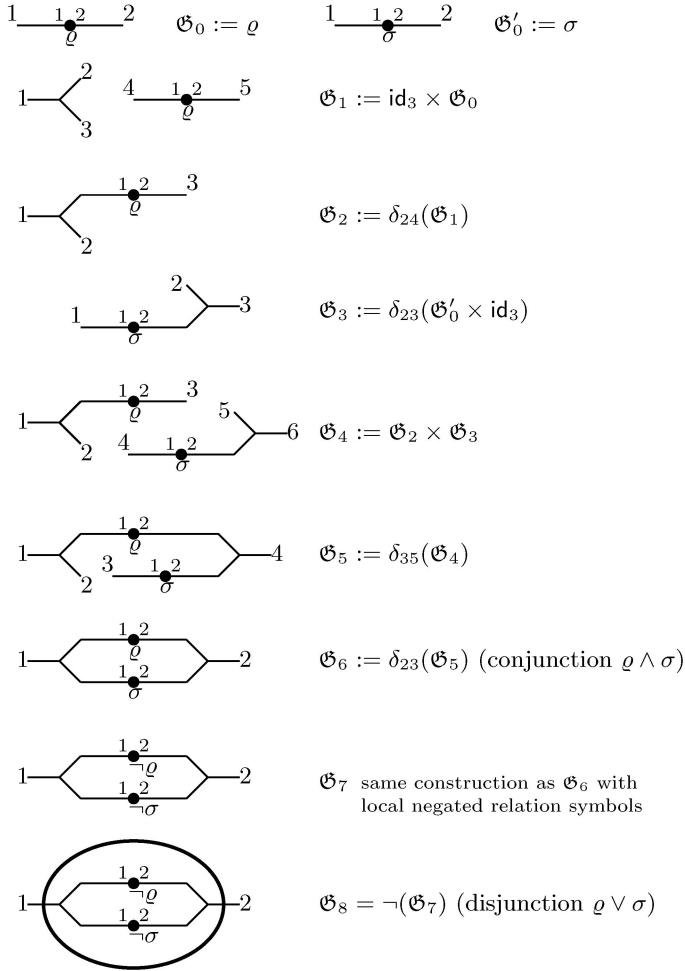


Fig. 4. Relation graphs and their description terms

Proof. (i): From the definitions (PAL1)–(PAL5) (Def. 7) follows immediately $\langle Q \rangle_{\text{PAL}} \subseteq \langle Q \rangle_{\text{RA}}$ (since $\langle Q \rangle_{\text{RA}}$ is closed under (R1) (Def. 2) and (R9), (R13), (R11) (Def. 3)). Conversely, every operation from (R1)–(R5) (Def. 2) is expressible by a PAL-construction (PAL1)–(PAL5) (Def. 7): in fact, $\Delta_A = \text{id}_2^A = \delta_{12}(\text{id}_1 \times \text{id}_3)^A$ (where $\text{id}_1 := \delta_{23}(\text{id}_3)$, see Fig. 3), ζ and τ are special cases of π_α , for the relational product we have $\varrho^A \circ \sigma^A = \delta_{m,m+1}(\varrho \times \sigma)^A$, and $\nabla \varrho^A = (\varrho \times \text{id}_1)^A$.

(ii) is just a special case of (i) taking into account local negation.

(iii): Note that negation \neg appears for Krasner algebras in (R7) (Def. 2) as well as for PAL in (PAL5) (Def. 7), and that union (R6) (Def. 2) can be expressed by negation and intersection: $\varrho \cup \sigma = \neg(\neg\varrho \cap \neg\sigma)$. Therefore (iii) immediately follows from (i). \square

Remark 10. As already mentioned in Example 8 the graphical presentation of relation graphs is very convenient. Because of Theorem 9 (iii) and Theorem 6 it is as powerful as the calculus of first order logic. Thus, relations graphs can be considered as a two-dimensional representation language for first-order formulas. Note further that – in the graphical presentation – one even need not worry about graph labels: a relation graph is just a collection of atomic relations (drawn in the plane like in Fig. 1) where pending edges may be connected in arbitrary way, and closed negation curves can be drawn in any appropriate way. Finally, to get a relation graph according to our definition, the remaining pending edges must be labelled by the first natural numbers $1, 2, \dots$ (recall, the relation labels belong to the nodes and their outgoing edges, they are needed if the relation graph is to be evaluated in a relational system over A and play no role during the construction of relation graphs).

3 The Teridentity

In this section we shall deal with the special role of the teridentity id_3 . From Theorem 9 (i),(iii) we get for $Q = \emptyset$ the set of all relations constructable in PAL without negation solely from the teridentity:

$$\langle \emptyset \rangle_{\text{pPAL}}^A = \langle \emptyset \rangle_{\text{RA}}, \quad \langle \emptyset \rangle_{\text{PAL}}^A = \langle \emptyset \rangle_{\text{KA}}.$$

It is known from e. g. [PösK79, 1.1.9(R1)] that $\langle \emptyset \rangle_{\text{RA}}$ equals the set D_A of all diagonal relations (see Def. 3 (R8)). Further, $\langle \emptyset \rangle_{\text{KA}}$ consists of all so-called pattern relations. A *pattern relation* $\varrho \in \text{Rel}^{(m)}(A)$ is a relation satisfying $\forall r, s \in A^m : r \in \varrho$ and $t(r) = t(s) \implies s \in \varrho$, where $t(r) := \{(i, j) \in \{1, \dots, m\}^2 \mid r(i) = r(j)\}$ is the *pattern* of an m -tuple in A^m .

As already mentioned the negated teridentity can simulate arbitrary negations; in fact we have:

Proposition 11. *For arbitrary finite A and arbitrary $Q \subseteq \text{Rel}(A)$ we have*

$$\langle Q \rangle_{\text{PAL}} = \langle Q \cup \{\neg(\text{id}_3^A)\} \rangle_{\text{pPAL}}.$$

Proof. It is known (e. g. [PösK79, 1.3.5]) that for $|A| \geq 3$ we have $\langle Q \rangle_{\text{KA}} = \langle Q \cup \{\nu^A\} \rangle_{\text{RA}}$ where $\nu^A := \{(a, b) \in A^2 \mid a \neq b\}$ is the inequality relation. As we have seen in Example 8 the inequality relation is the evaluation of the positive relation graph ν constructed with atomic relations id_3 and $\neg\text{id}_3$ in Fig. 3. Thus $\langle Q \rangle_{\text{PAL}} = \langle Q \rangle_{\text{KA}} = \langle Q \cup \{\nu^A\} \rangle_{\text{RA}} = \langle Q \cup \{\nu^A\} \rangle_{\text{pPAL}} \subseteq \langle Q \cup \{\neg\text{id}_3^A\} \rangle_{\text{pPAL}} \subseteq \langle Q \rangle_{\text{PAL}}$ and we are done.

The case $|A| = 2$ must be checked separately. We give only a sketch. It is straightforward to check that there is only one minimal clone of Boolean functions (on $A = \{0, 1\}$) preserving the negated teridentity $\neg\text{id}_3^A$, namely the clone $\text{Sym}(2)$ generated by the negation which consists up to fictitious variables only of the identity function and negation (conjunction, disjunction, constants, any majority or minority function do not preserve $\neg\text{id}_3^A$). The lattice of all Boolean clones

shows that therefore $\text{Sym}(2)$ is the only nontrivial clone preserving $\neg \text{id}_3^A$. Using some knowledge about the Galois closures of the Galois connections $\text{Pol} - \text{Inv}$ and $\text{Aut} - \text{Inv}$ (given by the formal context $(\text{Op}(A), \text{Rel}(A), \triangleright)$ where $f \triangleright \varrho: \iff f$ preserves ϱ) one can conclude $\langle Q \rangle_{\text{PAL}} = \langle Q \rangle_{\text{KA}} = \text{Inv Aut } Q = \text{Inv}(\text{Pol } Q \cap \text{Sym}(2)) = \text{Inv}(\text{Pol } Q \cap \text{Pol } \neg \text{id}_3^A) = \text{Inv Pol}(Q \cup \{\neg \text{id}_3^A\}) = \langle Q \cup \{\neg \text{id}_3^A\} \rangle_{\text{RA}} = \langle Q \cup \{\neg \text{id}_3^A\} \rangle_{\text{pPAL}}$. We do not go in more details here and refer to [PösK79, 1.2.3 and 1.3.5].

Remark 12. In the proof of Prop. 11 we have seen that there is a positive relation graph $\nu \in \langle \neg(\text{id}_3) \rangle_{\text{pPAL}}$ the evaluation of which gives the inequality relation ν^A on each set A (see Fig. 3, this even holds for infinite A). Proposition 11 implies that for every negated relation graph $\neg(\mathfrak{G}) \in \langle \Sigma \rangle_{\text{PAL}}$ and given finite base set A there exists a positive relation graph $\mathfrak{G}_1 \in \langle \Sigma \cup \{\neg \text{id}_3\} \rangle_{\text{pPAL}}$ with the same evaluation on A : $(\neg(\mathfrak{G}))^A = \mathfrak{G}_1^A$. However, contrarily to the inequality ν , this \mathfrak{G}_1 usually depends on A and cannot be chosen globally for all A . Moreover, the result does not hold in general for infinite A (here modifications are necessary, see also [Pös03]).

The next results show that the teridentity is – in some sense – indispensable as atomic relation in PAL. At first we show that it cannot be (positively) generated by relations of lower arity. To make this precise let us introduce the following notation: Let $\langle \Sigma \rangle_{\text{PAL} \setminus \{\text{id}_3\}}$ denote the set of all relation graphs constructed by (PAL1)–(PAL5) (Def. 7) but without using the teridentity id_3 in (PAL1) (analogously $\langle \Sigma \rangle_{\text{pPAL} \setminus \{\text{id}_3\}}$). Thus we have $\langle \Sigma \rangle_{\text{PAL}} = \langle \Sigma \cup \{\text{id}_3\} \rangle_{\text{PAL} \setminus \{\text{id}_3\}}$ and $D_A = \langle \text{id}_3^A \rangle_{\text{PAL} \setminus \{\text{id}_3\}}^A$.

Theorem 13. *Let A be a finite set with at least two elements and let $R_2 := \text{Rel}^{(1)}(A) \cup \text{Rel}^{(2)}(A)$. Then*



- (a) $\langle R_2 \rangle_{\text{PAL} \setminus \{\text{id}_3\}} \subsetneq \langle R_2 \rangle_{\text{PAL}} = \text{Rel}(A)$,
- (b) $\langle R_2 \rangle_{\text{pPAL} \setminus \{\text{id}_3\}} \subsetneq \langle R_2 \rangle_{\text{pPAL}} = \text{Rel}(A)$.

Remarks concerning the *Proof*:

The right-hand equality of (b) implies the corresponding equality in (a) and follows from the known fact $\langle \text{Rel}^{(2)}(A) \rangle_{\text{RA}} = \text{Rel}(A)$ (see e. g. [PösK79, 1.1.22]) using Theorem 9 (i). The inequality of (b) now follows immediately from (a).

It remains to prove the left-hand proper inclusion of (a). This can be done by showing $\text{id}_3^A \notin \langle R_2 \rangle_{\text{PAL} \setminus \{\text{id}_3\}}^A$. The proof is relatively technical and we do not have enough space for the full proof, therefore it is omitted here and we refer to [HerP].

Remark 14. Since the proof of Theorem 13 (a) was not given here, we add a proof of (b) which does not use (a). For (b), it is sufficient to show $\text{id}_3^A \notin \langle R_2 \rangle_{\text{pPAL} \setminus \{\text{id}_3\}}^A$. Assume on the contrary $\text{id}_3^A \in \langle R_2 \rangle_{\text{pPAL} \setminus \{\text{id}_3\}}^A$. Then there exists a ternary positive relation graph \mathfrak{G} (with three pending edges) such that every vertex has valency at most 2 and $\mathfrak{G}^A = \text{id}_3^A$. We choose \mathfrak{G} in such a way that it has a minimal number of vertices. But then there cannot be any non-pending edge

in \mathfrak{G} : in fact, if an edge connects two relations ϱ_1, ϱ_2 then this can be replaced by $\sigma = \varrho_1 \circ \varrho_2$; e. g. for binary ϱ_1, ϱ_2 , the graph  would be a subgraph of \mathfrak{G} and could be substituted by  such that the resulting graph has the same evaluation but less vertices in contradiction to vertex minimality of \mathfrak{G} . Consequently, \mathfrak{G} has only pending edges and therefore must be of the form $\mathfrak{G}_1 := \overset{1}{\sigma_1} 1 \overset{1}{\sigma_2} 2 \overset{1}{\sigma_3} 3$ or $\mathfrak{G}_2 := 1 \overset{1}{\varrho} 2 \overset{1}{\sigma} 3$ (here σ, σ_i are unary and ϱ is a binary relation symbol). But, whatever the relations $\sigma_i^A, \varrho^A, \sigma^A$ may be, neither $\mathfrak{G}_1^A = \sigma_1^A \times \sigma_2^A \times \sigma_3^A$ nor $\mathfrak{G}_2^A = \varrho^A \times \sigma^A$ equals the teridentity (provided that A has more than one element). \square

Remark 15. With Theorem 13 the first part of Peirce's thesis is proved, that we need (at least) ternary relations to build all relations from a set of basic relations.

The second part of the thesis is the inverse direction, that we can decompose any relation to ternary ones. According to Burch this is meant to know *which* relations (of small arity) construct a given relation. We refer for this to the theorem by Herzberger in [Her81]. This approach works if the domain A is sufficiently large, i. e. the cardinality of the domain is at least as large as the cardinality of the relation to be decomposed. This holds true for all relations in all infinite domains.

Burch proposes in [Bur91] a similar procedure for finite domains. However, to do this in general he allows to extend the underlying domain by new elements generated through the so-called *hypostatic abstraction*. While his explanations are philosophically of interest, such an extension to the underlying domain is uncommon in traditional mathematics. For this reason, we do not investigate this process in detail.

The next proposition also supports the special role of the teridentity: if a diagonal relation generates id_3 then it “contains” id_3^A (as projection), in particular, id_3^A is the only diagonal of minimal arity generating id_3 (and therefore generating all diagonal relations). The proof will be omitted here (and we refer to [HerP]).

Proposition 16. *Let A be a set with at least two elements. Then we have:*

- (a) $\text{id}_3^A \notin \langle \bigcup_{k=0}^{\infty} \text{Rel}^{(2k)}(A) \rangle_{\text{PAL} \setminus \{\text{id}_3\}}^A$,
- (b) $\text{id}_3^A \notin \langle \text{id}_1^A, \text{id}_2^A \rangle_{\text{PAL} \setminus \{\text{id}_3\}}$.
- (c) *Let $\varrho^A \in D_A$ be an m -ary diagonal relation. Then $\text{id}_3^A \in \langle \varrho^A \rangle_{\text{PAL} \setminus \{\text{id}_3\}}$ if and only if $m \geq 3$ is an odd number and there are $i, j, k \in \{1, \dots, m\}$ such that $\text{id}_3^A = \text{pr}_{ijk}(\varrho^A)$ (i. e. for $m=3$: $\varrho^A = \text{id}_3^A$).*

4 A Basic Generating System for PAL

The operations $\text{id}_3, \pi_\alpha, \times, \delta_{ij}, \neg$ of PAL are relatively easy to handle and intuitive. However, the operations π_α and δ_{ij} depend on the arity of the relations

(and therefore must be defined for every arity m yielding formally an infinite family of operations). For theoretical investigations a finite “minimal” system of operations is desirable. This can be done as in [PösK79] or [Arn01] for relational algebras. We mention here such a basic system for the construction of relation graphs (it reflects on the syntactical level most of the operations introduced in [Arn01]). The resulting set $\langle \Sigma \rangle_{\text{bPAL}}$ of relation graphs is formally a subset of $\langle \Sigma \rangle_{\text{PAL}}$ but nevertheless “the same” in the sense that for each $\mathfrak{G} \in \langle \Sigma \rangle_{\text{PAL}}$ there exists a $\mathfrak{B} \in \langle \Sigma \rangle_{\text{bPAL}}$ such that $\mathfrak{G}^A = \mathfrak{B}^A$ for every semantic domain A .

Definitions 17. Let \mathfrak{G} be an m -ary relation graph, $m \geq 2$. We introduce the following special PAL-operations:

- (PAL7) *basic rotation*: $\zeta(\mathfrak{G}) := \pi_\alpha(\mathfrak{G})$ where α is the cyclic permutation $1 \mapsto 2, \dots, m-1 \mapsto m, m \mapsto 1$ on $\{1, \dots, m\}$.
- (PAL8) *basic transposition*: $\tau(\mathfrak{G}) := \pi_\alpha(\mathfrak{G})$ where α is the transposition of the first two components: $1 \mapsto 2, 2 \mapsto 1$ and $i \mapsto i$ otherwise.
- (PAL9) *Connecting the first two pending edges*: $\Delta(\mathfrak{G}) := \delta_{12}(\mathfrak{G})$.

For $m \in \{0, 1\}$, i. e. if \mathfrak{G} is a unary or 0-ary relation graph, we set $\zeta(\mathfrak{G}) := \tau(\mathfrak{G}) := \Delta(\mathfrak{G}) := \mathfrak{G}$. The operations (PAL1) (atomic relations, including id_3), \times (PAL3), \neg (PAL5), ζ (PAL7), τ (PAL8) and Δ (PAL9) are called *basic* operations. The closure with respect to basic operation will be denoted by $\langle \Sigma \rangle_{\text{bPAL}}$, and as in Def. 7 we define $\langle Q \rangle_{\text{bPAL}} := \{\mathfrak{G}^A \mid \mathfrak{G} \in \langle \Sigma_Q \rangle_{\text{bPAL}}\}$ for a set $Q \subseteq \text{Rel}(A)$.

Formally we can state the following equality (the proof is in [HerP]):

Proposition 18. *For $Q \subseteq \text{Rel}(A)$ it holds $\langle Q \rangle_{\text{PAL}} = \langle Q \rangle_{\text{bPAL}}$.*

5 Contextual Logic and PAL

In this paper, the authors tried to harden the links between the Peircean Algebraic Logic and the realm of relational algebras from modern mathematics, while investigating some of Peirce’s claims regarding the special role of the teridentity.

However, the work on PAL has also to be considered in a broader context. On the one hand, as PAL is tightly connected with the Existential Graphs devised by Peirce, work on PAL is also of interest for the area of *Conceptual Graphs*, invented by John Sowa which are based on the Existential Graphs (see [Sow84, Sow92]).

The work on PAL has already influenced the work on the *Contextual Logic of Relations* as can be seen by the works [Wil00, Arn01, Pol02]. There a modification of PAL is used, which uses Power Context Families as semantic model of the relation graphs. Of course, the results of this paper can be easily transferred to this model. We hope that our work will contribute to the overall effort in this direction. Future research will concentrate on the investigation of deduction rules for the graphs introduced with PAL.

References

- [Arn01] M. ARNOLD, *Einführung in die kontextuelle Relationenlogik*. Diplomarbeit, Technische Universität Darmstadt, 2001.
- [Bur91] R. W. BURCH, *A Peircean Reduction Thesis: The Foundations of Topological Logic*. Texas Tech University Press, 1991.
- [Her81] HANS G. HERZBERGER, *Peirce's Remarkable Theorem*. In: L. W. SUMNER, J. G. SLATER, AND F. WILSON (eds.), *Pragmatism and Purpose: Essays Presented to Thomas A. Goudge*, University of Toronto Press, Toronto, 1981.
- [HerP] J. HERETH CORREIA AND R. PÖSCHEL, *Peircean Algebraic Logic and Relational Clones*. Manuscript in preparation.
- [Pol02] S. POLLANDT, *Relation graphs: a structure for representing relations in contextual logic of relations*. In: U. PRISS, D. CORBETT, AND G. ANGELOVA (eds.), *Conceptual Structures: Integration and Interfaces*, Proc. 10th Int. Conf. on Conceptual Structures, vol. 2393 of *LNAI*, Springer, 2002, pp. 34–47.
- [Pös03] R. PÖSCHEL, *Galois connections for operations and relations*. In: K. DENECHE, M. ERNÉ, AND S. WISMATH (eds.), *Galois connections and applications*, Kluwer, Dordrecht, 2003 (to appear).
- [PösK79] R. PÖSCHEL AND L. KALUŽNIN, *Funktionen- und Relationenalgebren*. Deutscher Verlag der Wissenschaften, Berlin, 1979, Birkhäuser Verlag Basel, Math. Reihe Bd. 67, 1979.
- [Sow84] J. F. SOWA: *Conceptual structures: Information processing in mind and machine*. Addison-Wesley, Reading 1984.
- [Sow92] J. SOWA, *Conceptual graphs summary*. In: T. NAGLE, J. NAGLE, L. GERHOLZ, AND P. EKLUND (eds.), *Conceptual structures: Current Research and Practice*, Ellis Horwood, 1992, pp. 3–52.
- [Tar41] A. TARSKI, *On the calculus of relations*. *J. Symbolic Logic* 6, (1941), 73–89.
- [Wil00] R. WILLE, *Lecture notes on Contextual Logic of Relations*. Preprint, Darmstadt University of Technology 2000.

Formal Concept Analysis for Knowledge Discovery and Data Mining: The New Challenges

Petko Valtchev¹, Rokia Missaoui², and Robert Godin³

¹ DIRO, Université de Montréal, Montréal (Qc), Canada

² Département d'informatique et d'ingénierie, UQO, Gatineau (Qc), Canada

³ Département d'informatique, UQAM, Montréal (Qc), Canada

Abstract. Data mining (DM) is the extraction of regularities from raw data, which are further transformed within the wider process of knowledge discovery in databases (KDD) into non-trivial facts intended to support decision making. Formal concept analysis (FCA) offers an appropriate framework for KDD, whereby our focus here is on its potential for DM support. A variety of mining methods powered by FCA have been published and the figures grow steadily, especially in the association rule mining (ARM) field. However, an analysis of current ARM practices suggests the impact of FCA has not reached its limits, i.e., appropriate FCA-based techniques could successfully apply in a larger set of situations. As a first step in the projected FCA expansion, we discuss the existing ARM methods, provide a set of guidelines for the design of novel ones, and list some open algorithmic issues on the FCA side. As an illustration, we propose two on-line methods computing the minimal generators of a closure system.

1 Introduction

Knowledge discovery in databases (KDD) is the process of discovering useful knowledge from data, e.g., within a data warehouse. Data mining (DM) is its main step, it consists in extracting potentially interesting regularities out of the initial data. According to Han et Kamber [14], the main challenges faced by DM researchers in the late 90s were the particularities of the target datasets, i.e., “large, noisy, of unknown generation laws, of high dimensionality, distributed”, that did not allow the conventional analysis methods to apply. Thus, the key features of DM tools and methods were efficiency and scalability to large datasets, robustness to missing and incorrect items in data, visualization and exploration of the mining results.

The core of FCA is an approach towards the design of conceptual hierarchies from a set of observations organized in a formal context. Therefore, the process of concept formation in FCA is a KDD *par excellence*, whereby the construction of the concept set constitutes the “mining” phase. The association rule mining (ARM) field has seen the largest number of successful methods that can be qualified as FCA-based [25, 42, 28]. A key advantage of the FCA-like mining lays in the fact that due to the closure properties only patterns of maximal size are extracted, thus reducing the exploration/interpretation burden for the analyst and increasing the overall efficiency [35].

However, the reduction in the pattern/rule number is only one particular benefit brought by the FCA framework. Indeed, nowadays, KDD, and ARM, in particular, faces

new challenges in more realistic situations, e.g., the dynamicity and the distribution of the datasets, the processing of rich data formats (XML documents, DNA sequences, OO data), etc. Therefore, the flexibility on changes in user settings and/or input data (i.e., the easy adaptation of pre-existing results to those changes), efficient assembly of partial views and mining results, tolerance to structure in the dataset, etc., become crucial features for mining tools.

We claim that FCA offers the theoretical constructs to meet the underlying challenges and that the issue is how these constructs are put behind fast algorithms. To support our claim, we put the existing “FCA-aware” work on ARM in the perspective of what are some open problems and new research directions. We then provide a set of new challenges together with their motivations and list the techniques that address them. These challenges are further translated into FCA algorithmic problems and, in some cases, solutions are proposed. We then focus on the on-line computation of the (minimal) generator family of a closure system, key task for the construction of specific non redundant bases of association rules. Two on-line algorithms are proposed that combine closure and generator maintenance and their practical performances are compared to batch ARM methods that also rely on generators.

The paper starts with a short recall on FCA theory and algorithms (Section 2). The relevant work on ARM with closed patterns and rule bases is then summarized followed by a discussion on the lessons that may be drawn from past experience (Section 3). We then suggest translations for the set of algorithmic problems into the FCA domain, before presenting recent algorithmic results on some of the identified problems (Section 4).

2 Background on Concepts, Lattices and Implications

2.1 FCA Basics

Formal concept analysis (FCA) [10] studies the way lattices¹ emerge out of data. Basic FCA considers an *incidence* relation I over a pair of sets O (*objects*, further denoted by numbers) and A (*attributes*, denoted by lower-case letters). The relation is given by the matrix of its incidence relation (oIa means that object o has the attribute a) which is called a (*formal*) *context* $\mathcal{K} = (O, A, I)$ (see Fig. 1, on the left). Following standard FCA notations, sets notations are separator-free, e.g., 127 stands for $\{1, 2, 7\}$. Moreover, I gives rise to two $'$ mappings: For a $X \subseteq O$, $X' = \{a \in A \mid \forall o \in X, oIa\}$, while $Y' = \{o \in O \mid \forall a \in Y, oIa\}$ for all $Y \subseteq A$ (e.g., $134' = fgh$ and $abc' = 127$ in Fig. 1). The pair of $'$ functions induces a *Galois connection* [3] between $\mathcal{P}(O)$ and $\mathcal{P}(A)$. The composite operators $''$ define *closures* on $\mathcal{P}(O)$ and $\mathcal{P}(A)$, hence each of them induces a family of *closed* subsets, denoted $\mathcal{C}_{\mathcal{K}}^o$ and $\mathcal{C}_{\mathcal{K}}^a$, respectively. A pair (X, Y) , of mutually corresponding subsets, i.e., $X = Y'$ and $Y = X'$, is called a (*formal*) *concept* in [36] whereby X is the *extent* and Y as the *intent* (e.g., $c = (134, fgh)$ is a concept in Fig. 1).

The set $\mathcal{C}_{\mathcal{K}}$ of all concepts of \mathcal{K} is partially ordered by extent inclusion: $(X_1, Y_1) \leq_{\mathcal{K}} (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2 (Y_2 \subseteq Y_1)$. In fact, provided with \subseteq , $\mathcal{C}_{\mathcal{K}}^o$ and $\mathcal{C}_{\mathcal{K}}^a$ become

¹ An excellent introduction to partial orders and lattices may be found in [7].

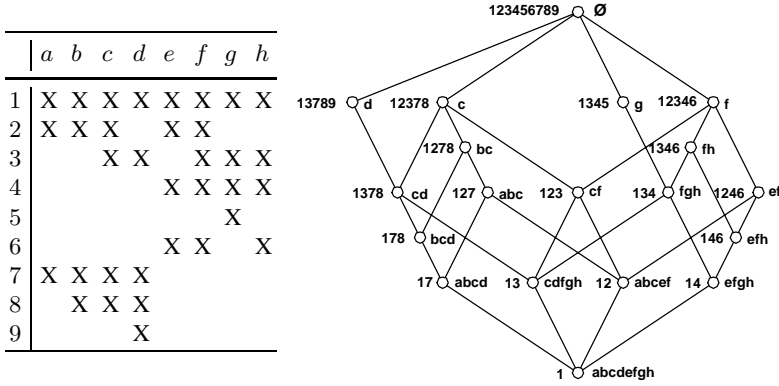


Fig. 1. Left: Context \mathcal{K} (adapted from [10]) with $O = \{1, 2, \dots, 9\}$ and $A = \{a, b, \dots, h\}$. **Right:** The Hasse diagram of the concept (Galois) lattice derived from \mathcal{K} .

two *complete lattices* which are dually isomorphic via $'$. These lattices overlap perfectly, thus giving rise to the *concept lattice*² $\mathcal{L}_{\mathcal{K}} = \langle \mathcal{C}_{\mathcal{K}}, \leq_{\mathcal{K}} \rangle$ of the context \mathcal{K} [10]. Moreover, in $\mathcal{L}_{\mathcal{K}}$ the joins are $\bigvee_{i=1}^k (X_i, Y_i) = ((\bigcup_{i=1}^k X_i)'', \bigcap_{i=1}^k Y_i)$ and meets $\bigwedge_{i=1}^k (X_i, Y_i) = (\bigcap_{i=1}^k X_i, (\bigcup_{i=1}^k Y_i)'')$. Fig. 1 on the right shows the Hasse diagram of $\mathcal{L}_{\mathcal{K}}$ (e.g., $(123, cf) \vee (1246, ef) = (12346, f)$ and $(123, cf) \wedge (1246, ef) = (12, abcefg)$).

2.2 Implications

Dependencies among attributes in the dataset constitute important type of knowledge and may be the goal of a separate analysis process. FCA offers a compact representation mode for attribute dependencies, the *implication rules*, with two sets, *premise* and *conclusion*, $X \rightarrow Y$ ($X, Y \subseteq A$).

An implication is *valid* in a context if none of the objects violates it: $X \rightarrow Y$ is valid iff $Y \subseteq X''$ (e.g., $cg \rightarrow h$ is valid in \mathcal{K} from Fig. 1, whereas $cd \rightarrow f$ is not). Moreover, a rule is *informative* [13] if its premise is minimal and its consequence maximal for set inclusion (e.g., $ab \rightarrow c$, $a \rightarrow bc$ are both valid but only the later is informative). The minimal valid premises for a given conclusion are called (minimal) *generators* (or *key sets*). Formally, $Y \subseteq A$ is a generator iff $\forall \tilde{Y} \subset Y, \tilde{Y}'' \subset Y''$.

The set $\Sigma_{\mathcal{K}}$ of all valid implications in a context \mathcal{K} may be very large and restricting it to valid rules may not help much. In contrast, subsets of much smaller size, or *bases*, encode the same information content as $\Sigma_{\mathcal{K}}$ without a loss. The content is retrieved through inference mechanisms for rule sets, such as the one due to Armstrong, the so called *Armstrong axioms* [18].

For example, the *Duquenne-Guigues basis* [13], $\mathcal{B}_{\mathcal{K}}$, is known to be minimal in the number of rules. It relies on *pseudo-closed* sets: $Y \subseteq A$ is pseudo-closed if it is not closed and for any other pseudo-closed Z , $Z \subset Y$ entails $Z'' \subset Y$ [10]. $\mathcal{B}_{\mathcal{K}}$ is

² Also known as the *Galois lattice* [3]), it appeared in the work of Öre [21] and Birkhoff [4].

made out of all the rules $Y \rightarrow Y''$ where Y is pseudo-closed. The entire set $\Sigma_{\mathcal{K}}$ may be obtained as the *implicational hull* of $\mathcal{B}_{\mathcal{K}}$.

Implication rules as closely related to *functional dependencies* in the database field (see [18]) made their way into data mining (see Section 3). In fact, approximative associations correspond to *partial*, i.e., not necessarily overall valid, implications, whereas exact associations are the counterpart of (valid) implications. For example, the rule $ab \rightarrow cd$ is valid to 66,67%. In [17], a basis for partial implications is provided, later called the *Luxemburger cover basis*. It is made out of all rules of the form $\bar{Y} \rightarrow Y - \bar{Y}$ where \bar{Y} and Y are closed attribute sets (intents) and (Y'', Y) is a lower cover of (\bar{Y}'', \bar{Y}) in the respective lattice. For example, the basis of the context in Fig. 1 includes $c \rightarrow b$ and $c \rightarrow d$, both of 80% validity.

2.3 Lattice Construction

A variety of algorithms exist for constructing the lattice of a context [9, 5, 12, 20]. We only recall key algorithms while interested readers are referred to [16]. A classical distinction between lattice algorithms is made upon two axes: computation of the lattice order (precedence) relation and evolution of the context during construction.

Batch Construction. The algorithm NEXTCLOSURE designed by Ganter [9] uses a lexicographic order on concept intents to avoid redundant concept generations without a global memory of the already generated concepts.

Batch algorithms constructing both concepts and order have been proposed first by Bordat [5] and then by Nourine and Raynaud [20]. The first algorithm uses structural properties of the precedence relation in \mathcal{L} to generate concepts from their upper covers. The second one uses a general incremental procedure for constructing closed/open set families to generate the concepts and cardinality properties for extents of neighbor concepts to retrieve the precedence links in the lattice.

On-Line Construction. On-line (incremental) algorithms are designed to fit evolution in data. Classical object incremental methods [12] construct the lattice $\mathcal{L}_{\mathcal{K}}$ starting from a single object o_1 and gradually incorporating any new object o_i into the lattice \mathcal{L}_{i-1} (over a context $\mathcal{K}_{i-1} = (\{o_1, \dots, o_{i-1}\}, A, I_{i-1})$ where mappings $'$ are denoted $^{i-1}$ whenever confusion may occur). At each step, a set of structural updates are carried out locally and at a limited points of the lattice, while keeping the rest unchanged. Thus, the available structure is used to limit the search/reconstruction effort relevant to a single insertion.

The basic property underlying the paradigm (see [33] for a detailed description) states that $\mathcal{C}_{\mathcal{K}}^a$ is closed under intersection, thence the increase of \mathcal{K}_{i-1} potentially adds new intents to it through intersection with o'_i (function $\mathcal{Q} : \mathcal{C}_{i-1} \rightarrow 2^A$, with $\mathcal{Q}(X, Y) = Y \cap \{o\}'$). Thus, the concepts whose intents are in $\mathcal{Q}(\mathcal{C}_{\mathcal{K}_{i-1}}) - \mathcal{C}_{\mathcal{K}_{i-1}}^a$ are called *new* (denoted $\mathbf{N}^+(o_i)$). Two further sorts of concepts are looked for in \mathcal{L}_{i-1} by the method: The *modified* concepts, $\mathbf{M}(o_i)$, have their intents in $\mathcal{Q}(\mathcal{C}_{\mathcal{K}_{i-1}}) \cap \mathcal{C}_{\mathcal{K}_{i-1}}^a$. The *genitor*³ concepts (X, Y) , $\mathbf{G}(o_i)$, previously called *generators* (term changed here to

³ The American Heritage Dictionary of the English Language: Fourth Edition. 2000: **genitor**: 1. One who produces or creates. 2. *Anthropology* A natural father or mother.

avoid ambiguity), obey to: $Y \not\subseteq o_i$ and $Y = \mathcal{Q}(X, Y)''$. Let now mapping ς simulate the way \mathcal{L}_{i-1} “evolves” to \mathcal{L}_i , i.e., ς is mapping from \mathcal{L}_{i-1} to \mathcal{L}_i based on intent preservation. While the counterparts of modified concepts (X, Y) have their extents changed to $(X \cup \{o_i\})$, i.e., $\varsigma(X, Y) = (X \cup \{o_i\}, Y)$, on the remainder of $\mathcal{C}_{\mathcal{K}_{i-1}}$ ς follows identity, i.e., $\varsigma(X, Y) = (X, Y)$. The images of both distinguished concept sets are denoted $\mathbf{G}^+(o_i) = \mathbf{G}(o_i)$ and $\mathbf{M}^+(o_i) = \mathbf{M}(o_i)$. Finally, genitors “seed” the creation of new concepts via the bijection $\gamma : \mathbf{G}(o_i) \rightarrow \mathbf{N}^+(o_i)$ with $\gamma(X, Y) = (X \cup \{o\}, Y \cap o')$.

Given a lattice \mathcal{L} and a new object o , the construction of the augmented lattice \mathcal{L}^+ amounts to detecting $\mathbf{G}(o)$ and $\mathbf{M}(o)$, creating $\mathbf{N}^+(o)$ and modifying $\mathbf{M}(o)$, then adjusting the precedence relation. The set of tasks, structured into a generic scheme in [31], is summarized in Algorithm 1. The approach relies on a characterization of both $\mathbf{G}(o)$ and $\mathbf{M}(o)$ as the unique maxima of the respective equivalence classes in \mathcal{L} induced by \mathcal{Q} . Moreover, a key fact about the precedence relation in \mathcal{L}^+ says that when connecting a new concept c to its neighbor concepts in \mathcal{L}^+ , the only lower cover of c which comes from \mathcal{L} is the image of respective genitor $\varsigma(\gamma^{-1}(c))$. More generally, among non-new concepts, only genitors *have* their upper covers changed, and only modified *may have* their lower covers changed. Consequently, given a concept c from $\mathcal{L}^+ - \mathbf{N}^+(o)$ and a new concept c_n from $\mathbf{N}^+(o)$, $c \leq_{\mathcal{L}^+} c_n$ if and only if $c \leq_{\mathcal{L}^+} \varsigma(\gamma^{-1}(c))$.

```

1: procedure ADD-OBJECT(In/Out:  $\mathcal{L} = \langle \mathcal{C}, \leq \rangle$  a lattice; In:  $o$  an object)
2:
3: for all  $c$  in  $\mathcal{C}$  do
4:   if  $c = \max([c]_{\mathcal{Q}})$  then
5:     if  $\text{Intent}(c) \subseteq o'$  then
6:        $\text{Extent}(c) \leftarrow \text{Extent}(c) \cup \{o\}$ 
7:     else
8:        $\hat{c} \leftarrow \text{NEW-CONCEPT}(\text{Extent}(c) \cup \{o\}', \mathcal{Q}(c)) ; \mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$ 
9:       UPDATE-ORDER $(\hat{c}, c)$ 
```

Algorithm 1. Generic scheme for the insertion of a new object into a concept (Galois) lattice.

As databases evolve by adding/deleting sets of objects rather than single ones, in [32] we generalized the incremental paradigm to context subposition and the corresponding assembly of factor lattices.

2.4 Context Splits and Lattice Assembly

Subposition of Contexts and Semi-product of Lattices. Subposition is the horizontal assembly of contexts⁴ sharing the same set of attributes [10]. Let $\mathcal{K}_1 = (O_1, A, I_1)$ and $\mathcal{K}_2 = (O_2, A, I_2)$ be two contexts sharing the attribute set A , the context $\mathcal{K}_3 = (O_1 \dot{\cup} O_2, A, I_1 \dot{\cup} I_2)$ is called their *subposition*, denoted $\mathcal{K}_3 = \frac{\mathcal{K}_1}{\mathcal{K}_2}$. For example, for the context $\mathcal{K} = (O, A, I)$ as given in Fig. 1, let $O_1 = \{1, 2, 3, 4\}$ and $O_2 = \{5, 6, 7, 8, 9\}$. The factor lattices corresponding to \mathcal{K}_1 and \mathcal{K}_2 , say \mathcal{L}_1 and \mathcal{L}_2 , are given in Fig. 2.

⁴ Apposition is dual assembly upon O ; both generalize to n -splits.

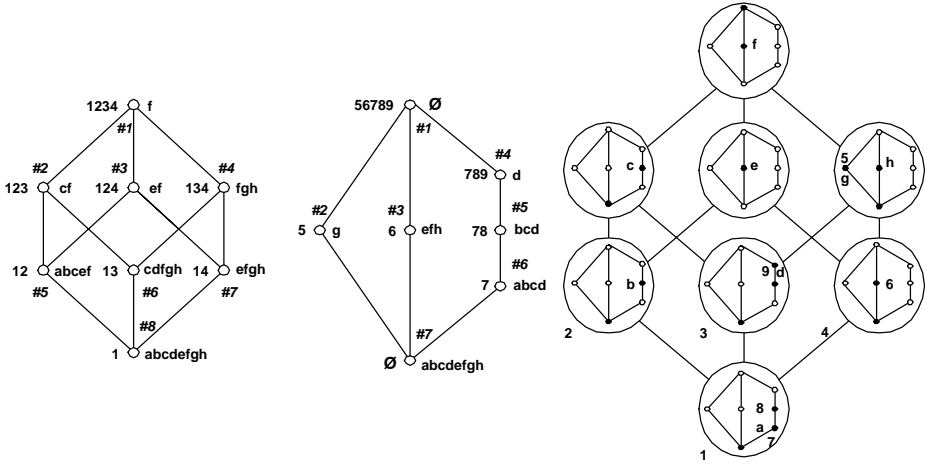


Fig. 2. Left: Factor lattices \mathcal{L}_1 and \mathcal{L}_2 of the context in Fig. 1. Right: The NLD of \mathcal{L}_3 .

The lattices \mathcal{L}_1 and \mathcal{L}_2 are related to the lattice of the subposition context, called the *semi-product* in [10], \mathcal{L}_3 , in a specific way. In the extreme case, \mathcal{L}_3 will be isomorphic to the direct product of \mathcal{L}_1 and \mathcal{L}_2 , $\mathcal{L}_1 \times \mathcal{L}_2$ (denoted shortly $\mathcal{L}_{1,2}$ in the sequel). However, in the general case, \mathcal{L}_3 is only a meet sub-semi-lattice of $\mathcal{L}_{1,2}$.

Furthermore, two mappings link \mathcal{L}_1 and \mathcal{L}_2 to \mathcal{L}_3 . The composition of factor concepts into a global one is made along the intent dimension shared by \mathcal{K}_j ($j = 1, 2, 3$): the semi-product operation may be seen as the merge of two closure spaces on A . Each node $((X_1, Y_1), (X_2, Y_2))$ from $\mathcal{L}_{1,2}$ is sent to a concept (X, Y) from \mathcal{L}_3 such that $Y = Y_1 \cap Y_2$ (e.g., in Fig. 2, $(c_{\#7}, c_{\#3})$ is sent to $(146, efh)$). The resulting mapping from $\mathcal{L}_{1,2}$ to \mathcal{L}_3 is a surjective order morphism that preserves lattice joins (see [34] for details). Conversely, \mathcal{L}_3 is mapped onto \mathcal{L}_j ($j = 1, 2$) by simply projecting concept intents on A_j (e.g., $(127, abc)$ is projected to the node $(c_{\#5}, c_{\#6})$).

Merge of Factor Lattices. A framework for the effective computation of the semi-product was studied in [34, 32]. The factor merge basically filters $\mathcal{L}_{1,2}$, and keeps only the nodes from the meet sub-semi-lattices isomorphic to \mathcal{L}_3 . These nodes, like in the singleton case, are the canonical members of their respective equivalence classes (upon intersection of factor intents). Indeed, as the function that maps nodes from $\mathcal{L}_{1,2}$ to concepts from \mathcal{L} is not injective, a further property states that $((X_1, Y_1), (X_2, Y_2))$ with $X = X_1 \cup X_2$, is the maximum in the class where $Y = Y_1 \cap Y_2$ and $X = Y^3$.

Canonical node definition together with a characterization of the precedence relation in \mathcal{L}_3 underly the straightforward procedure for lattice merge illustrated by Algorithm 2.

The canonicity test `CANONICAL()` is based on a comparison of intent intersection on a node (c_i, c_j) (variable I) to the intersection on its upper covers in $\mathcal{L}_{1,2}$. For example, $(c_{\#7}, c_{\#3})$ is canonical since the intersection of factor extents, efh , is strictly greater

```

1: procedure MERGE(In:  $\mathcal{L}_1, \mathcal{L}_2$  lattices; Out:  $\mathcal{L}_3$  a lattice)
2:
3:  $\mathcal{L} \leftarrow \emptyset$ 
4:  $\text{SORT}(\mathcal{C}_1); \text{SORT}(\mathcal{C}_2) \{\text{Decreasing order}\}$ 
5: for all  $(c_i, c_j)$  in  $\mathcal{L}_1 \times \mathcal{L}_2$  do
6:    $I \leftarrow \text{Intent}(c_i) \cap \text{Intent}(c_j)$ 
7:   if  $\text{CANONICAL}((c_i, c_j), E)$  then
8:      $c \leftarrow \text{NEW-CONCEPT}(\text{Extent}(c_i) \cup \text{Extent}(c_j), I)$ 
9:      $\text{UPDATE-ORDER}(c, \mathcal{L}_3)$ 
10:     $\mathcal{L}_3 \leftarrow \mathcal{L}_3 \cup \{c\}$ 

```

Algorithm 2. Assembling the global Galois lattice from a pair of partial ones.

than the intersections on immediate successors $(c_{\#7}, c_{\#1})$, $(c_{\#3}, c_{\#3})$ and $(c_{\#4}, c_{\#3})$ (\emptyset , ef , and fh , respectively).

To construct the lattice of a context, Algorithm 2 is called recursively on ever smaller fragments of the context, going down to a single column of the binary table.

3 Association Rule Mining Problem

Let $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$ be a set of m distinct items. A transaction T contains a set of items in \mathcal{I} , and has an associated unique identifier called *TID*. A subset Y of \mathcal{I} where $k = |Y|$ is referred to as a k -itemset (or simply an itemset). A transaction database (TDB), say \mathcal{D} , is a set of transactions. The fraction of transactions in \mathcal{D} containing Y is the support of Y , $\text{supp}(Y)$, given either as an absolute count or as a percentage (e.g., $\text{supp}(bcd)$ in the TDB represented by the context in Fig. 1 is 33%). An itemset is frequent (or large) when $\text{supp}(Y)$ reaches at least a user-specified minimum threshold called *minsupp*.

Consider the context in Fig. 1 which may be thought of as a sample set of transactions from a retailer database $\mathcal{D} = \{1, \dots, 9\}$ involving items $\mathcal{I} = \{a, \dots, h\}$. The itemsets Y with $\text{supp}(Y) \geq 40\%$ are given hereafter.

Itemset	Supp.	Itemset	Supp.	Itemset	Supp.	Itemset	Supp.
b	4	c	5	d	5	f	5
g	4	h	4	e	4	-	-
bc	4	cd	4	fh	4	ef	4

3.1 Association Rule Generation

An association rule is an implication of the form $X \rightarrow Y$, where X and Y are subsets of \mathcal{I} , and $X \cap Y = \emptyset$ (e.g., $d \rightarrow c$). The support of a rule $X \rightarrow Y$ is defined as $\text{supp}(X \cup Y)$ while its confidence is computed as the ratio $\text{supp}(X \cup Y) / \text{supp}(X)$. For example, the support and confidence of $d \rightarrow c$ are 44% and 80% respectively.

The problem of mining association rules with given minimum support and confidence (called *minconf*) can be split into two steps: (i) detection of all frequent (large)

itemsets (*FIs*) X , i.e., those with $\text{supp}(X) \geq \text{minsupp}$, and (ii) generation of the association rules r with $\text{conf}(r) \geq \text{minconf}$. It is admitted that (ii) is relatively straightforward once the result of (i) is there. However, (i) presents a considerable challenge because of the potential combinatorial explosion on top of \mathcal{I} .

3.2 The Frequent Closed Itemset Framework

Since the most time consuming operation in association rule generation is the computation of frequent itemsets, some recent studies have proposed a search space pruning based on the computation of frequent *closed* (*FCIs*) itemsets only, without any loss of information. In particular, FCA-powered approaches have been suggested to that end in [41, 25]. The gain is in producing and storing only a subset of the *FIs* without a loss of information since non closed *FIs* can be easily obtained from closed ones. More specifically, a key result states that any itemset has the same support as its closure (follows from the properties of the $'$ operators in FCA), hence it is equally frequent. Besides, in [26], an itemset X is considered as closed if it cannot be increased without reducing the support: $\forall i \in \mathcal{I} - X, \text{supp}(X \cup \{i\}) < \text{supp}(X)$. The subset of the concept set corresponding to the *FCIs* is usually referred to as the *iceberg* (concept) lattice [28]. This is clearly equivalent to the definition given in Section 2.

As indicated earlier, association rules can be advantageously generated from *FCIs* rather than *FIs*. However, *FCIs* do not decrease the number of generated rules on their own. A step further is the production of non-redundant rule sets, or bases.

A first basis relies on the notion of *generator* of a closed itemset [25, 27] that was discussed in Section 2.2. A **generic basis** for exact association rules is a collection of rules of the form: $Z \rightarrow Z'' - Z$ such that Z is a generator for Z'' and $Z \neq Z''$. The generic basis can hence be constructed from *FCIs* and generators only (e.g., (178, *bcd*) in Fig. 1 leads to the generation of rule $bd \rightarrow c$ with a support of 44%).

A similar basis is defined for approximative association rules, in particular as a remedy of the problem of informativeness in the Luxenburger basis. Indeed, the rules in the latter basis are not necessarily *maximally* informative (i.e., the premise is *minimized* and the consequence is *maximized*). Thus, the informative basis has been defined in [23], where every rule is maximally informative. An **informative basis** is a set of rules of the form: $Z \rightarrow Y_2 - Z$ where Z is a generator for Y_1 such that $C_1 = (X_1, Y_1)$, $C_2 = (X_2, Y_2)$ and C_1 covers C_2 . Support and confidence of a rule r in such basis are $\text{supp}(Y_2)$ and $|Y_2'|/|Z'|$ respectively. Based on Fig. 1, $b \rightarrow ac$ ($\text{supp} = 44\%$, $\text{conf} = 75\%$) and $fg \rightarrow eh$ ($\text{supp} = 33\%$, $\text{conf} = 67\%$) belong to the informative basis.

3.3 Computation of FCI Families and Rule Bases

Similarly to lattice algorithms, ARM methods are divided into batch and on-line, whereby on-line here covers a broader data evolution scenarios. Moreover, target structures may differ: *FCI*, complete sets of rules, rule bases, etc. Finally, the input format is a criterion too: standard itemsets are opposed to sequences of items or itemlists or to even richer descriptions.

FCI Miners. Historically, the reference FI mining algorithm is APRIORI [1]. It performs a level-wise generation of FIs within the powerset lattice 2^T , starting by singleton sets and moving upwards and level-wise in 2^T . At each level, the candidates are generated by joining FIs from the previous level that differ by a single element. Candidates which have at least one non-frequent subset are pruned *a priori*, i.e., without looking at the database to estimate frequencies.

ACLOSE and CLOSE [24] probably the first FCI miners. Like APRIORI, it performs level-wise computation within the powerset lattice, but this time based on the generators of the FCIs. Generators replace candidates in the APRIORI framework; they guide the FCI look-up in the database. TITANIC [28] is a descendent of ACLOSE, but relies on advanced features of generators to avoid redundant computation, e.g., cardinality reasoning for closure computation and minimality tests for the filtering of non-generator sets.

CHARM [42] is another closed pattern miner which generates FCIs in a tree organized by inclusion. Closure and support computation relies on storage and intersection of *TID-sets* (i.e., the set of transactions per item, the equivalent of concept extent). To speed-up closure computation, it uses *diffsets*, the set difference on the TID-sets of a given node and of its unique parent node in the tree.

CLOSET and its recent modification CLOSET+ [35] both generate FCIs as maximal branches of a *FP-tree*, a structure that is basically a prefix tree (or *trie*) augmented with transversal lists of pointers. The global FP-tree of a database is projected into a set of conditional FP-trees that organize patterns sharing the same suffix. Cardinality reasoning is applied to compute the closure of a given branch in the FP-tree.

Computation of various rule bases has been addressed in [25] and in [41]. However, to the best of our knowledge, there is no efficient method for their construction. An interesting insight on rule bases and their effective construction is provided in [15].

Finally, a very recent trend in closure-based mining is the discovery of closed patterns on richer descriptions, i.e., patterns that range over a collection of structured objects such as sequences [22, 39] or graphs [38] and that represent sub-objects common to some of the collection members.

Advanced FI Mining Paradigms. Realistic databases often evolve in time, with regular insertion and removal of bunches of transactions. On-line mining algorithms were introduced to cope with data evolution at low cost, i.e., without starting from scratch.

Early incremental FI miners were based on the APRIORI framework. FUP [6] updates the set of association rules whenever some new transactions are added. The candidates for the incremental transaction set are generated with respect to their frequencies in the initial database which are in turn deduced from some pre-computed support information for the database. The descendant of the FUP method, FUP-2 admits a larger set of operations on the database, including insertion, removal and modification of transactions. An alternative paradigm for on-line FI mining relies on the notion of *negative border* [19], i.e., the set of all infrequent itemsets that are minimal for inclusion (see [8] and [29] for concrete methods). More recently, the UWEP [2] method was proposed which performs a look-ahead pruning to drop any unfrequent candidate as early as possible.

To the best of our knowledge, the problem of on-line FCI mining has not been addressed within the data mining community. This is mainly due to the fact that along the data evolution, patterns may repeatedly change their status between frequent and infrequent. The underlying difficulties are easily summarized in the one-case, i.e., where a single transaction is added: given the initial and the target families of FCIs, some new FCIs may be produced by genitors that are not themselves frequent enough and therefore lay out of the initial iceberg. Thus, the classical incremental reasoning about genitors and modified concepts holds no more.

Parallelism and distribution constitute a yet different computation paradigm that has been largely explored in data mining, in particular for cost reduction purposes (see [40] for a survey). It is noteworthy that most of the strategies for parallel mining rely on a distribution of the dataset among the available processors.

Observations. Despite the apparent proximity between the respective targets, algorithms for lattice/implication basis construction and association miners are based on diverging principles which impede easy adaptation of FCA methods to the mining tasks. To summarize the distinctions, one could say that ARM methods are designed to work on very large datasets that do not hold in the main memory of a computer and therefore keep the access to the raw data (in the database) to a minimum. For instance, CLOSET only scans the database twice, whereas TITANIC would make a number of scans that equals the size of the largest generator. In contrast, NEXTCLOSURE will look at the data table on each closure computation, i.e., a prohibitively large number of times.

Moreover, the potentially huge size of the mining results limits the quantity of information that can be stored per FI or FCI. Typically, the itemsets are stored with their support, a number, and rules with their premises, conclusion, support and confidence. In contrast, beside concept intents, lattice algorithms may store extents and order as well, mainly because these are relied on at a later step of the calculations. Space limits forced the ARM methods to use advanced data structures that enable speedy computation of the basic operations, i.e., set inclusion and intersection, as well as direct access to sub-patterns. For example, prefix-tree-like structures, e.g., FP-trees, are massively used in the compact storage of itemset families, whereas hash tables provide direct access to sets of patterns.

3.4 Lessons Learned

Based on our experience both in FCA and KDD, we strongly believe that FCA is a suitable paradigm for KDD [37] since it offers valuable features such as the strong mathematical background, the availability of algorithmic methods that can perform conceptual clustering, implication and association rule generation, the effective framework for data fragmentation (e.g., apposition, concatenation) and the reverse fusion together with the corresponding lattice operations, etc. Moreover, line diagrams and scaling can be seen as tools for visualization/browsing and data preprocessing mechanisms, respectively. Moreover, when focusing on the data mining step and on the particular activity of ARM, basic theoretical results from FCA have been successfully used to reduce the size of the output in both ARM tasks, i.e., FI mining and rule extraction. However, the work

done by the FCA community [13, 11, 17, 23] do not seem to have a significant echo in the DM field. This is partly due to the fact that the proposed FCA algorithms do not compete with the leading algorithms in a particular subfield, or, in some cases, to the lack of sufficient empirical evidence on how those methods scale over large datasets.

We firmly believe that the potential of FCA is much bigger than the current status, in particular in what concerns handling of data evolution, distributive and parallel computation, and coping with structure in the data items. To put it in a more fancy way, in order to attract stronger interest from the general KDD and DM community, FCA should (also) stand for **F**acility, **C**ost-effectiveness and **A**daptability.

Facility is the ease of use, in particular on various data formats (expressiveness). This aspect covers both the availability of user-friendly FCA tools and the compatibility with various mining settings such as constraints, multi-level conceptual hierarchies, or other structure defining domain knowledge.

Cost-effectiveness amounts to insuring high performance and good scalability. It can be achieved through an effective and efficient implementation of DM algorithms, and, whenever applicable, a well-organized decomposition of the mining effort using parallel/distributed architectures.

Adaptability is the faculty to easily change or be changed in order to fit evolving situations such as modified user needs, changes in input data, new system constraints. To reach adaptability, FCA should offer on-line mining (e.g., incremental procedures for adding/removing transactions or itemsets), and provide mechanisms for adapting the DM step to the needs of the user.

In the following section, we present a set of results intended to clear the way for new FCA-based mining methods that fit the above requirements.

4 Flexible Mining of FCIs and Association Rule Bases

4.1 Mining Scenarios and Related FCA Problems

As a starting point, we have identified a set of mining scenarios that correspond to practical situations and considered the underlying algorithmic problems. The following is a first attempt to draw a systematic list of the FCA problems whose solution is of potential interest to the data mining community. In fact, Table 1 presents each situation with the respective operations on contexts, CI/FCI families, and rule bases, whereby algorithmic problems are given unique identifiers pX .

Before going on, it is noteworthy to underline the fact that in order to compute the various bases, one needs the concept intents, the pseudo-closed (Duquenne-Guigues), the generators (generic, informative), and order relation (Luxenburger).

We have paid significant attention to the problem $p1$, i.e., the incremental maintenance of the FCI family upon insertions of single transaction. The problem is the basic building block for the study of larger increments, on the one hand, and is close in spirit to the incremental construction of the lattice, on the other hand. However, as noted in the previous section, substantial differences exist between input structures, in particular in the availability of genitor/modified information for all new concepts that need to be created. Indeed, in the iceberg case, there might be new FCIs in the target structure whose genitors are “hidden” below the “sea-level”, i.e., the border of the iceberg.

Table 1. The translation of mining scenarios into algorithmic problems.

Scenario	Context	CI/FCI	Rule bases
ADD transac- tions	Subposition	Increment/Assembly ($p1/p2$) of FCI families on ground set A	Merge of Duquenne-Guigues ($p3$), Luxenburger cover ($p4$), informa- tive/generic ($p5$) bases
REMOVE transactions	Horizontal Split	Contraction of an FCI family ($p2'$)	Filtering of Duquenne-Guigues ($p3'$), Luxenburger cover ($p4'$), informa- tive/generic ($p5'$) bases
JOIN database fragments (views)	Apposition	Assembly ($p6$) of dis- joint FCI families	Merge of disjoint Duquenne-Guigues ($p7$), Luxenburger cover ($p8$), infor- mative/generic ($p9$) bases
PROJECT a ta- ble on a subset of attributes	Vertical Split	Projection of an FCI family ($p6'$)	Factoring of Duquenne-Guigues ($p7'$), Luxenburger cover ($p8'$), informative/generic ($p9'$) bases

Clearly, the discovery of those new FCIs by an updating procedure cannot rely on the standard genitor/modified framework.

The problem of the incremental FCI computation was therefore approached with strategies relying on the computation of all the CIs of the database. The approach effectively removes the obstacle created by hidden genitors/modified, but the price to pay is the storage of the entire CI family whereas usually only a tiny part of it will be examined by the analyst. The extra storage increases substantially memory consumption and algorithmic cost of the method. To avoid processing the entire concept set, two separate techniques have been applied (see [33]). In the first method, GALICIA-P, an index (item \times CI) is used to avoid producing all the the empty intersections between an existing concept intent and the new object description. The second algorithm, GALICIA-LBU, uses a bottom-up lattice traversal which is not typical for object increments. Thus, it looks for a quick jump from any concept which is not maximal in its class $\llbracket Q$, to the effective maximum of that class. The jump mechanism has its own cost since it forces the lattice order to be available and hence to be maintained. When compared to CLOSET, the former algorithm has shown satisfactory performances.

Another sort of problems that we have studied carefully is the maintenance of implication bases upon the insertion of one new object. The following section explains how generators may be efficiently extracted upon single-object increments and how the corresponding procedure generalizes to the lattice assembly case. In contrast, the computation of all pseudo-closed in the incremental settings, has not been tackled so far because of its inherent complexity.

Problems related to object removal (REMOVE line of the above table) have the opposite effect on the mining results when compared to their ADD counterpart. Consequently, the respective methods could simply reverse the incremental scheme. Therefore, we do not insist on the theoretical foundations of the removal operations.

In the dual case, i.e., with attributes evolving, the reasoning is unfortunately not symmetric. Indeed, the computation of generators is much tricky in the attribute case than in the object case. However, the iceberg maintenance does not present serious challenges in both cases. Moreover, the Duquenne-Guigues basis may be obtained in

an indirect manner, as shown in [30]. The underlying algorithm produces both closed and pseudo-closed sets. Like in the object case, the removal problems just follow the reverse scheme of their ADD counterpart, that is why we do not focus on them here.

4.2 On-Line Computation of the Generator Family for \mathcal{C}^a

Definitions and Notations. Given a context $\mathcal{K} = (O, A, I)$, consider the family of its concept intents, \mathcal{C}^a , ordered by inclusion and the corresponding equivalence relation induced by the associated closure operator \prime . The latter will be denoted explicitly $\varphi_{\mathcal{C}^a}$ in order to avoid confusion.

Our first goal is to clarify the evolution of the set of generators in every concept from \mathcal{C} along the transition from \mathcal{L} to \mathcal{L}^+ . To denote the various generator sets, we shall use the following notations:

- $gen : \mathcal{P}(\mathcal{P}(A)) \rightarrow \mathcal{P}(\mathcal{P}(A))$ assigns the set of all generators to a given family,
- $gen_{\mathcal{C}^a} : \mathcal{C} \rightarrow \mathcal{P}(\mathcal{P}(A))$ assigns to a closed set in a family \mathcal{C}^a , the set of its generators, i.e., the minima of its equivalence class,
- $\Delta gen_{\mathcal{C}_1^a \rightarrow \mathcal{C}_2^a}$ is a shortcut for $gen_{\mathcal{C}_1^a}(Y) - gen_{\mathcal{C}_2^a}(Y)$.

Structural Results. The generator evolution is first tackled in a global manner. Thus, we consider the status of a generator from $gen(\mathcal{C}^a)$ in \mathcal{C}^{a+} . A first result states that whenever Y is a generator in \mathcal{C}^a , its status either remains steady or it can change so that Y becomes a closed set in \mathcal{C}^{a+} . In the latter case, Y is a *new* closed set, i.e., corresponds to the intent of a concept from $\mathbf{N}^+(o)$. More generally speaking, Y has a different closure in \mathcal{C}^+ , i.e., $\varphi_{\mathcal{C}^a}(Y) \neq \varphi_{\mathcal{C}^{a+}}(Y)$. The examination of all the cases of sets changing their closures after o has been inserted, leads to the observation that the old closure is necessarily the intent of a genitor concept and the new one is the intent of a new concept.

Lemma 1 *For a set $Y \subseteq A$, whenever its closure changes in \mathcal{C}^{a+} , i.e., $\varphi_{\mathcal{C}^a}(Y) \neq \varphi_{\mathcal{C}^{a+}}(Y)$, the closed sets involved may be characterized as follows:*

1. $\varphi_{\mathcal{C}^a}(Y)$ is the intent of a concept from $\mathbf{G}(o)$,
2. $\varphi_{\mathcal{C}^{a+}}(Y)$ is the intent of a concept from $\mathbf{N}^+(o)$.

Further to the above property, one may express the evolution of the entire equivalence classes from \mathcal{C}^a to \mathcal{C}^{a+} . In fact, the equivalence class of a new intent is exactly the part of the class of its genitor intent which is made out of elements that are themselves included in the new intent. Conversely, the new equivalence class of a genitor is included in the set difference between the old class and the elements mentioned before, i.e., all those included in the respective new intent. In all other cases, the class remains the same both in \mathcal{C}^a and \mathcal{C}^{a+} .

Corollary 1 *For any $c \in \mathcal{C}^{a+}$, the following holds:*

$$\begin{aligned}
 c \in \mathbf{G}^+(o) : & \quad [Int(c)]_{\mathcal{C}^{a+}} \subseteq [Int(c)]_{\mathcal{C}^a} - \mathcal{P}(Int(\gamma(c))) \\
 c \in \mathbf{N}^+(o) : & \quad [Int(c)]_{\mathcal{C}^{a+}} = [Int(\gamma^{-1}(c))]_{\mathcal{C}^a} \cap \mathcal{P}(Int(c)) \\
 else : & \quad [Int(c)]_{\mathcal{C}^{a+}} = [Int(c)]_{\mathcal{C}^a}
 \end{aligned}$$

Another remarkable fact is that any attribute set Y which is minimal in its respective class $[Y]_{\mathcal{C}^a}$ is still minimal in its new class $[Y]_{\mathcal{C}^{a+}}$.

Lemma 2 *For all $Y \in A$, $Y \in \min([Y]_{\mathcal{C}^a})$ implies $Y \in \min([Y]_{\mathcal{C}^{a+}})$.*

It is noteworthy that the above lemma holds even in case of a non-closed Y becoming closed in \mathcal{C}^{a+} . As a result, one may assert that all the generators in \mathcal{C}^a either remain generators in \mathcal{C}^{a+} or become closed sets, i.e., members of \mathcal{C}^{a+} .

Corollary 2 $gen(\mathcal{C}^a) \subseteq gen(\mathcal{C}^{a+}) \cup \mathcal{C}^{a+}$.

Now that we know that all the generators can only become closed or stay generators, the complementary question comes to the light: where do new generators come from?

First, it is noteworthy that only in the case of a genitor concept, new generators can emerge in \mathcal{C}^{a+} . In fact, as the elements in the new classes in \mathcal{C}^{a+} , i.e., those corresponding to new intents, preserve their inclusion-based order, any minimal element Y of such a class is minimal in its class in \mathcal{C}^a . Thus, to characterize the new generators, one has to focus on the minima of the genitor classes in \mathcal{C}^{a+} . The real difference is then a sum of all differences between actual and all minima over the set of all genitors. Obviously, $gen(\mathcal{C}^{a+}) - gen(\mathcal{C}^a) = \cup_{c \in \mathbf{G}^+(o)} \Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$, where, following previous results, $\Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ may be written as $(\min([Int(c)]_{\mathcal{C}^a} - \mathcal{P}(Int(\gamma(c)))) - \min([Int(c)]_{\mathcal{C}^a}))$. We shall now characterize the sets $\Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ where c is in $\mathbf{G}^+(o)$. In fact, we show that all new generators come from former generators for the same c to which an attribute from the difference between the genitors intent and the new intent $(Int(c) - Int(\gamma(c)))$ is added. This difference is called the “face” [27] of $Int(c)$ with respect to $Int(\gamma(c))$.

Property 1 *For any $Y \in \Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ for a given c from \mathcal{L} where c is a genitors, the following holds:*

$$Y = Y_o \cup \{a\}$$

where $Y_o \in \min([Int(\gamma(c))]_{\mathcal{C}^{a+}})$ and $a \in Int(c) - Int(\gamma(c))$

This property states that every new generator in $\Delta gen_{\mathcal{C}^{a+} \rightarrow \mathcal{C}^a}(c)$ is the union of a generator that went to $\gamma(c)$ and an attribute that belongs to $Int(c)$ but not to $Int(\gamma(c))$.

We illustrate our approach, we follow the evolution of the equivalence class associated with the closet itemset $cdfgh$ attached to the genitor $(13, cdfgh)$ is depicted in Figures 3. As it may be seen in left part of the figure, the generators of $bedgh$ before the insertion of object 7 are d , cg , and ch . Once object 7 is added, a new concept $(137, cd)$ is derived from the genitor $(13, cdfgh)$ and the generator d moves to the class of its intent cd (see the right part of the figure). This migration leaves three new minima in the class of $cdfgh$: df , dg , and dh . Thus, the set of all generators associated to $cdfgh$ in the new family \mathcal{C}^{a+} increases to $\{cg, ch, df, dg, dh\}$.

4.3 A Scheme for Incremental Generator Construction

The structural results from the previous paragraphs underlie a procedure (see Algorithm 3) which, given a generator concept c and its corresponding new concept \bar{c} computed using Algorithm 1, updates the set of generators associated with the intent of c and identifies the set of generators attached to the intent of \bar{c} .

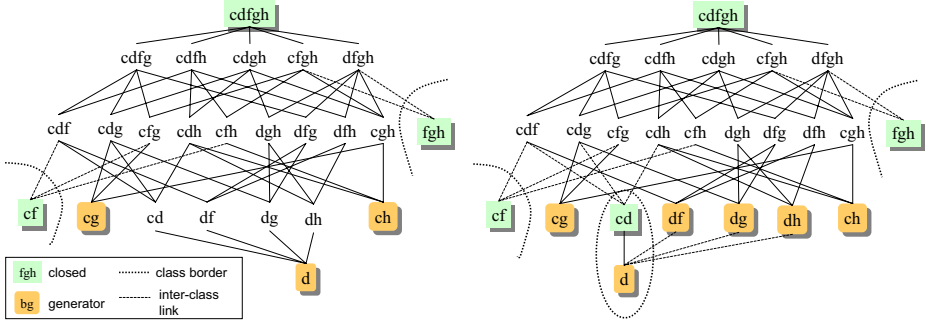


Fig. 3. The equivalence class of the closet set $cdfgh$ in 2^A prior to the insertion of object 7 ($abcd$) into the context \mathcal{K}_1 (left) and after the insertion (right).

```

1: procedure COMPUTE-GENERATORS(In/Out:  $c, \bar{c}$  concepts)
2:
3: for all  $g$  in  $c.gens$  do
4:   if  $g \subseteq \bar{c}.Intent$  then
5:      $\bar{c}.gens \leftarrow \bar{c}.gens \cup \{g\}$ 
6:    $c.gens \leftarrow c.gens - \bar{c}.gens$ 
7: SORT( $\bar{c}.gens$ )
8: for all  $\bar{g}$  in  $\bar{c}.gens$  do
9:    $new-gens \leftarrow \emptyset$ 
10:  for all  $a$  in ( $c.Intent - \bar{c}.Intent$ ) do
11:     $gen-cond \leftarrow \text{true}$ 
12:    for all  $g$  in  $c.gens$  do
13:      if  $g \subseteq \bar{g} \cup \{a\}$  then
14:         $gen-cond \leftarrow \text{false}$ 
15:      if  $gen-cond$  then
16:         $new-gens \leftarrow new-gens \cup \{\bar{g} \cup \{a\}\}$ 
17:   $c.gens \leftarrow c.gens \cup new-gens$ 

```

Algorithm 3. Computation of the generators of a new concept and of its genitor.

The proposed procedure for generator extraction relies on the properties of a generator [27] of a closet itemset as well as the structural results we defined in the previous section.

The algorithm includes two main tasks: (i) identify the generators of the intent of a new concept \bar{c} from the generators g in $c.gens$ (i.e., the generators of the CI related to c , the genitor of \bar{c}), and (ii) update the generators of the CI related to concept c by discarding any generator g that is included in the intent of \bar{c} and augmenting any generator \bar{g} (identified at the first step) with individual items a from $c.Intent - \bar{c}.Intent$ whenever the produced itemset is minimal (i.e., the condition of line 13 does not hold).

4.4 Implementation and Performance Tests

The INC-GEN algorithm was implemented in Java, within the 1.1 version of the Galicia platform⁵.

⁵ See the website at: <http://www.iro.umontreal.ca/~galicia>.

The method has been tested as a stand-alone application and its performance was compared to those of two basic algorithms for FCI mining that rely on generator computation, CLOSE and ACLOSE. The experiments were done on a Windows PC station (Pentium III 996 MHz with 512 MB of RAM) using various subsets of the IBM transaction database T25I10D10K. This dataset is made out of 10 000 transactions over a set of 10 000 items. It is known to be a sparse one, with an average of 25 items per transaction. The graphs drawn in Fig. 4 summarize our findings so far. They clearly indicate

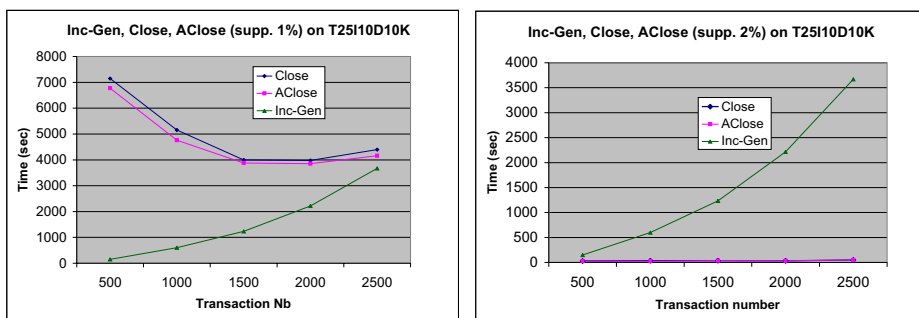


Fig. 4. Evolution of the CPU-time for all three algorithms on transaction batches up to 2500 drawn from the dataset T25I10D10K. CLOSE and ACLOSE were given min. supp. thresholds of 1% (left) and 2% (right).

that the incremental method, INC-GEN, is not competitive as a batch miner for support thresholds of 2% and up. This fact is not surprising given the large number of concepts that the algorithm must examine on each insertion of a new transaction and the even larger number of generators. For example, the first 2 500 transactions produce 900 000 generators in the entire concept set. It is more surprising to see that for supports of 1% and lower, the incremental method scores better than the batch ones, at least for the first quarter of the dataset. This result is even more surprising given the large discrepancy in the number of closed sets and generators produced (about 3 000 for CLOSE/ACLOSE versus more than 900 000 for INC-GEN) and the fact that the overwhelming part of the CPU time is spent on generator computation.

Further tests will be necessary to fully understand the behavior of the INC-GEN method. However, the current track seems promising, especially if combined with an efficient on-line miner of iceberg lattices.

4.5 Generator Evolution along the Factor Lattice Assembly

Generator computation can easily extend to the construction of the subposition-based semi-product of factor lattices \mathcal{L}_1 and \mathcal{L}_2 . First, recall that any concept in the semi-product \mathcal{L}_3 is created by a pair of factor concepts that play symmetric role (called the genitors). We nevertheless adopt an asymmetric view on factors and set \mathcal{L}_1 to the initial lattice where generators already exit whereas \mathcal{L}_2 is seen as the surrogate for “new” concepts that constitute \mathcal{L}_3 . Consequently, when such a new concept is detected by the

assembly algorithm, its generators will be computed with respect to its genitor in \mathcal{L}_1 , i.e., the respective component of the canonical representative in $\mathcal{L}_{1,2}$. Obviously, unlike the object-wise increment, there can be several new concepts per genitor: these are exactly the concepts $c_3 = (X, Y)$ from \mathcal{L}_3 where $Y^{11} = \text{Intent}(c_1)$ ⁶. The challenge will be to determine their generators without an interference between those.

Next, observe that new concepts corresponding to a genitor c_1 have intents that lay in the equivalence class $[\text{Intent}(c_1)]_{11}$. Moreover, they define a partition of this class into finer classes according to the ³³ closure, whereas a unique new concept has the same intent as c_1 .

A safe strategy for consecutive insertions of the new concepts is to insure that whenever a new c_3 is inserted, there is a larger intent above $\text{Intent}(c_3)$ in the class $[\text{Intent}(c_1)]_{11}$ whose generators are known and can be filtered in the way described in Algorithm 3. The straightforward way to do this is to perform insertions in an order compatible with \leq_3 , i.e., starting with smaller intents and then proceeding with larger ones. Given the set of the intents of new concepts generated by c_1 , say $\mathcal{C}_3^a \cap [\text{Intent}(c_1)]_{11}$, the previous condition imposes that at any time the set of already inserted concepts corresponds to an order ideal of that set, provided with inclusion order.

A noteworthy fact about the concrete computation method is that it perfectly fits Algorithms 2 and 3 (with parameters c_1 and c_3). Moreover, all along the insertions, the temporary set of generators that are to be considered for the next insertion is stored at the genitor node within \mathcal{L}_3 . Indeed, the concept corresponding to c_1 in \mathcal{L}_3 (i.e., with the same intent) will be the last one to be created since its intent is the greatest element of the equivalence class. For example, the evolution of the equivalence class associated to $cdfgh$ (from \mathcal{L}_1) is depicted in Fig. 5. Indeed, the inner loop of Algorithm 2 discovers three new concepts with intents d , cd , and $cdfgh$, respectively, and in this order. These are gradually “inserted” in the class of $cdfgh$: the generators of the new intent are computed and those of $c_1 = (13, cdfgh)$ are updated in \mathcal{L}_1 . Thus, the new intent d which corresponds to an initial generator of the class forces the creation of four new generators (cd , df , dg , dh). In contrast, the closed cd merely converts a former generator into a closure.

5 Conclusion

A fundamental problem with association rule mining (ARM) is the enormous amount of information that needs to be managed. FCA has already had an important impact on this problem with the introduction of FCI mining and related minimal covers for rules including the Duquenne-Guigues, generic, Luxenberger and informative bases. Several important algorithmic contributions are rooted in these concepts.

Improvement to the flexibility of ARM can be achieved through approaches that adapt to dataset evolution. Incremental approaches for mining CI, FCI and related bases is one direction that we have tackled with some success. Furthermore, this work has revealed some important insights on the properties of these objects from an evolutionary point of view including the more general divide and conquer context fusion problem. Another direction that could contribute to the flexibility of ARM tools is the adaptation

⁶ Recall that Y^{ii} is the closure of Y in \mathcal{K}_i .

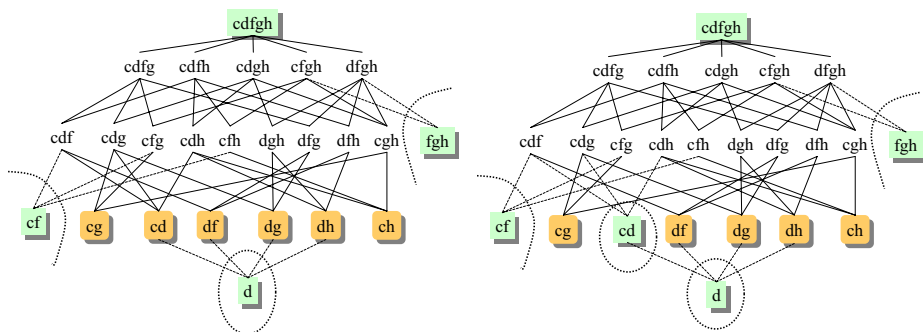


Fig. 5. The evolution of the equivalence class of the closet set $cdfgh$ in $\mathcal{P}(A)$ during the assembly process (see initial state in Fig. 3, on the left): after the creation of the new closed d (left) and after the creation of cd (right).

to user needs. In the same spirit as OLAP analysis tools, the incremental and fusion approaches are also relevant to this aspect of the problem by providing techniques to dynamically handle several levels of details in the analysis process as expressed by user needs.

Finally, a direction that could contribute to the flexibility of ARM tools but that has had little impact up to now is the ability to handle more expressive data representations (many-valued contexts, objects, scaling, etc.). Much work remains to be done in order to fully exploit the power of FCA in the overall knowledge discovery process.

Acknowledgments

This research was supported by the authors' individual NSERC grants as well as by the FQRNT team grant.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago, Chile, September 1994.
2. N. Ayan, A. Tansel, and M. Arkun. An efficient algorithm to update large itemsets with early pruning. In *Proceedings, KDD-99*, pages 287–291, San Diego (CA), USA, 1999. ACM Press.
3. M. Barbut and B. Monjardet. *Ordre et Classification: Algèbre et combinatoire*. Hachette, 1970.
4. G. Birkhoff. *Lattice Theory*, volume XXV of *AMS Colloquium Publications*. AMS, 3rd edition, 1967.
5. J.-P. Bordat. Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques et Sciences Humaines*, 96:31–47, 1986.
6. D. W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. In *Proceedings, ICDE-96*, pages 106–114, New Orleans (LA), USA, 1996.

7. B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 1992.
8. R. Feldman, Y. Aumann, A. Amir, and H. Mannila. Efficient Algorithms for Discovering Frequent Sets in Incremental Databases. In *Proceedings, ACM SIGMOD Workshop DMKD'97*, pages 59–70, Tucson (AZ), USA, 1997.
9. B. Ganter. Two basic algorithms in concept analysis. preprint 831, Technische Hochschule, Darmstadt, 1984.
10. B. Ganter and R. Wille. *Formal Concept Analysis, Mathematical Foundations*. Springer-Verlag, 1999.
11. R. Godin and R. Missaoui. An Incremental Concept Formation Approach for Learning from Databases. *Theoretical Computer Science*, 133:378–419, 1994.
12. R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11(2):246–267, 1995.
13. J.L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Sociales*, 95:5–18, 1986.
14. J. Han and M. Kamber. *Data Mining : Concepts and Techniques*. Morgan Kaufmann, 2001.
15. M. Kryszkiewicz. Concise representations of association rules. *Pattern Detection and Discovery*, pages 92–109, 2002.
16. S. Kuznetsov and S. Ob'edkov. Comparing the performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):189–216, 2002.
17. M. Luxenburger. Implications partielles dans un contexte. *Mathématiques et Sciences Humaines*, 29(113):35–55, 1991.
18. D. Maier. *The theory of Relational Databases*. Computer Science Press, 1983.
19. H. Mannila, H. Toivonen, and A. Verkamo. Efficient algorithms for discovering association rules. In U. Fayyad and R. Uthurusamy, editors, *Proceedings, AAAI Workshop on Knowledge Discovery in Databases*, pages 181–192, Seattle (WA), USA, 1994. AAAI Press.
20. L. Nourine and O. Raynaud. A Fast Algorithm for Building Lattices. *Information Processing Letters*, 71:199–204, 1999.
21. O. Öre. Galois connections. *Transactions of the American Mathematical Society*, 55:493–513, 1944.
22. F. Pan, G. Cong, A. Tung, J. Yang, and M. Zaki. Carpenter: Finding closed patterns in long biological datasets. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington (DC), August, 2003.
23. N. Pasquier. Extraction de bases pour les règles d'association à partir des itemsets fréquents. In *Proceedings of the 18th INFORSID'2000*, pages 56–77, Lyon, France, 2000.
24. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings, ICDT-99*, pages 398–416, Jerusalem, Israel, 1999.
25. N. Pasquier, Y. Bastide, T. Taouil, and L. Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems*, 24(1):25–46, 1999.
26. J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Proceedings, ACM SIGMOD Workshop DMKD'00*, pages 21–30, Dallas (TX), USA, 2000.
27. J. Pfaltz and C. Taylor. Scientific discovery through iterative transformations of concept lattices. In *Proceedings of the 1st International Workshop on Discrete Mathematics and Data Mining*, pages 65–74, Washington (DC), USA, April 2002.
28. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing Iceberg Concept Lattices with Titanic. *Data and Knowledge Engineering*, 42(2):189–222, 2002.
29. S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases. In *Proceedings, KDD-97*, pages 263–266, New Port Beach (CA), USA, 1997.

30. P. Valtchev and V. Duquenne. Towards scalable divide-and-conquer methods for computing concepts and implications. In E. SanJuan, A. Berry, A. Sigayret, and A. Napoli, editors, *Proceedings of the 4th Intl. Conference Journées de l'Informatique Messine (JIM'03): Knowledge Discovery and Discrete Mathematics, Metz (FR), 3-6 September*, pages 3–15. INRIA, 2003.
31. P. Valtchev, M. Rouane Hacene, and R. Missaoui. A generic scheme for the design of efficient on-line algorithms for lattices. In B. Ganter A. de Moor, W. Lex, editor, *Proceedings of the 11th Intl. Conference on Conceptual Structures (ICCS'03)*, volume 2746 of *Lecture Notes in Computer Science*, pages 282–295, Berlin (DE), 2003. Springer-Verlag.
32. P. Valtchev and R. Missaoui. Building concept (Galois) lattices from parts: generalizing the incremental methods. In H. Delugach and G. Stumme, editors, *Proceedings of the ICCS'01*, volume 2120 of *Lecture Notes in Computer Science*, pages 290–303, 2001.
33. P. Valtchev, R. Missaoui, R. Godin, and M. Meridji. Generating Frequent Itemsets Incrementally: Two Novel Approaches Based On Galois Lattice Theory. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):115–142, 2002.
34. P. Valtchev, R. Missaoui, and P. Lebrun. A partition-based approach towards building Galois (concept) lattices. *Discrete Mathematics*, 256(3):801–829, 2002.
35. J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington, DC, USA, 2003.
36. R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470, Dordrecht-Boston, 1982. Reidel.
37. R. Wille. Why can concept lattices support knowledge discovery in databases. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):81–92, 2002.
38. X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington (DC), 2003.
39. X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. In R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *Proceedings of the 3rd SIAM International Conference on Data Mining (ICDM'03)*, San Fransisco (CA), 2003.
40. M.J. Zaki. Parallel and Distributed Association Mining: A Survey. *IEEE Concurrency*, 7(4):14–25, december 1999.
41. M.J. Zaki. Generating Non-Redundant Association Rules. In *Proceedings, KDD-00*, pages 34–43, Boston (MA), USA, 2000.
42. M.J. Zaki and C.-J. Hsiao. CHARM: An Efficiently Algorithm for Closed Itemset Mining. In R. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *Proceedings of the 2nd SIAM International Conference on Data Mining (ICDM'02)*, 2002.

AddIntent: A New Incremental Algorithm for Constructing Concept Lattices

Dean van der Merwe¹, Sergei Obiedkov², and Derrick Kourie¹

¹ Department of Computer Science, University of Pretoria
Pretoria, 0002, South Africa

Deon.vd.Merwe@bentleywest.com, DKourie@cs.up.ac.za

² Institute für Algebra, TU Dresden, 01062 Dresden, Germany
s-obj@yandex.ru

Abstract. An incremental concept lattice construction algorithm, called AddIntent, is proposed. In experimental comparison, AddIntent outperformed a selection of other published algorithms for most types of contexts and was close to the most efficient algorithm in other cases. The current best estimate for the algorithm's upper bound complexity to construct a concept lattice L whose context has a set of objects G , each of which possesses at most $\max(|g'|)$ attributes, is $O(|L||G|^2 \max(|g'|))$.

1 Introduction

In Formal Concept Analysis (FCA) [1, 2], the problem of generating the set of all concepts of a formal context and constructing the diagram graph (covering relation) of the concept lattice has been well studied [3–7]. See [8] for an overview and comparison. Since lattices can grow exponentially large, leading to correspondingly large computational requirements, efficient algorithms for the construction of concept lattices are of key importance.

In this text, we introduce a new lattice construction algorithm, called AddIntent, an earlier version of which appeared in [9] under the name AddAtom. The algorithm produces not only the concept set, but also the diagram graph. Being incremental, it relies on the graph constructed from the first objects of the context to integrate the next object into the lattice. Therefore, its use is most appropriate in those applications that require both the concept set and diagram graph, for example, in applications related to information retrieval and document browsing [10]. Nevertheless, experiments show that sometimes it takes other algorithms longer merely to construct the concept set of a context than it takes AddIntent to construct the entire diagram graph of the same context.

The upper bound complexity estimate of the algorithm is linear in the lattice size modulo some factor polynomial in the input size. Although the current estimate of the polynomial factor is higher than that of the lowest known for lattice construction algorithms (i.e., that of [11]), experimental comparison indicates that, in practice, AddIntent performs well in constructing lattices from many different contexts. It outperforms other algorithms in most cases and may therefore be considered as the best candidate

for a universal lattice construction algorithm for diagram graphs amongst the algorithms considered in this study.

2 Main Definitions

This section defines the basic terminology and notation of Formal Concept Analysis [1].

Definition 1. A formal context is a triple of sets (G, M, I) , where G is called a set of objects, M is called a set of attributes, and $I \subseteq G \times M$. The notation gIm indicates that $(g, m) \in I$ and denotes the fact that the object g possesses the attribute m .

Definition 2. For $A \subseteq G$ and $B \subseteq M$:

$$A' = \{m \in M \mid \forall g \in A(gIm)\} \text{ and } B' = \{g \in G \mid \forall m \in B(gIm)\}.$$

The operator $'$ is a closure operator (to be precise, $'$ is a homonymous denotation of two closure operators: $2^G \rightarrow 2^G$ and $2^M \rightarrow 2^M$).

Definition 3. A formal concept of a formal context (G, M, I) is a pair (A, B) , where $A \subseteq G, B \subseteq M, A' = B$, and $B' = A$. The set A is called the extent, and the set B is called the intent of the concept (A, B) .

For $g \in G$ and $m \in M$, $\{g\}'$ is denoted by g' and called an object intent, and $\{m\}'$ is denoted by m' and called an attribute extent.

Definition 4. For a context (G, M, I) , a concept $X = (A, B)$ is less general than or equal to a concept $Y = (C, D)$ (or $X \leq Y$) if $A \subseteq C$ or, equivalently, $D \subseteq B$.

Definition 5. For two concepts X and Y , if $X \leq Y$ and there is no concept Z with $Z \neq X, Z \neq Y, X \leq Z \leq Y$, the concept X is called a lower neighbour (or a child) of Y and Y is called an upper neighbour (or a parent) of X . This relationship is denoted by $X \prec Y$.

Definition 6. We call the (directed) graph of the relation \prec a diagram graph. A plane embedding of a diagram graph where a concept has larger vertical coordinate than that of any of its lower neighbors is called a line (Hasse) diagram.

3 The AddIntent Algorithm

In this section, we define the AddIntent algorithm and describe the basic strategy it follows. Readers are referred to [12] for a more detailed discussion.

AddIntent is an incremental algorithm, i.e., it takes as input the lattice L_i produced by the first i objects of the context and inserts the next object g to generate a new lattice L_{i+1} . As observed in [5, 7] lattice construction can be described in terms of four sets of concepts: *modified concepts*, *generator concepts*, *new concepts* and *old concepts*.

A concept $(C, D) \in L_{i+1}$ is *new* if D is not an intent of any concept in L_i . We call a concept $(A, B) \in L_i$ *modified* if $B \subseteq g'$ since g has to be added to its extent in L_{i+1} . Otherwise, $B \cap g' = D \neq B$ for some concept $(C, D) \in L_{i+1}$. If (C, D) is a new concept, then (A, B) is called a *generator* of (C, D) ; if not, (A, B) is called *old*.

Every new concept (C, D) has at least one generator, but it may have several. The (unique) most general of these generators is called the *canonical generator* of (C, D) . The remaining generators of (C, D) are naturally called its *non-canonical* generators. Obviously, there is a one-to-one correspondence between canonical generators and new concepts.

The key problem of incremental construction is how to identify all modified concepts (in order to add g to their extents) and all canonical generators of new concepts (in order to generate every new concept exactly once). Efficient algorithms will spend as little effort as possible searching through the unmodified and non-canonical generators. AddIntent approaches this problem by traversing the diagram graph of L_i in a recursive fashion.

There are several ways to identify canonical generators. The algorithm in [5] processes the list of concepts in L_i starting with the most general ones. In processing a concept (A, B) , it generates the intent $B \cap g'$ and then looks through the set of new concepts produced so far (and arranged in “buckets” according to the cardinality of their intents) to see whether such an intent is already there. The algorithm of Norris [6], in processing a concept (A, B) , checks whether $B \cap g' \subseteq h'$ for any $h \in Gi \setminus A$ (assuming that G_i is the set of objects processed up to that moment); if so, then A is not the maximal extent, and hence (A, B) is not the most general concept capable of generating $B \cap g'$. What both algorithms ignore are the following facts:

Proposition 1. *If (B', B) is a canonical generator of a new concept (F', F) , while (D', D) is a non-canonical generator of (F', F) – in this case, $B \subset D$ – then any concept (H', H) such that $H \subset D$ and $H \not\subset B$ is neither modified nor is it a canonical generator of any new concept.*

Proposition 2. *If (D', D) is an old concept and $D \cap g' = B$ – in this case, $(B', B) \in L_i$ is modified – then any concept (H', H) such that $H \subset D$ and $H \not\subset B$ is neither modified nor is it a canonical generator of any new concept.*

Therefore, there is no need to process such concepts (H', H) in the search of canonical generators and modified concepts. Since AddIntent maintains the diagram graph (which explicitly orders concepts from most to least general) it can exclude these concepts from further consideration by simply traversing the diagram graph in a bottom-up fashion. Having found a non-canonical generator, AddIntent uses the diagram graph to find the canonical generator of the same concept. It then works only with concepts above that canonical generator, ignoring all other concepts above the non-canonical generator. The canonical generator can be found in a diagram graph in at most $O(|G|^2|M|)$ time.

These ideas are expanded in [12] where the notions of the *approximate intent representative* and *exact intent representative* are introduced in the context of so-called compressed pseudo-lattices.

Using parameter names to imply types the *AddIntent* function is defined as follows:

```

01: Function AddIntent(intent, GeneratorConcept, L)
02:   GeneratorConcept = GetMaximalConcept(intent,
                                           GeneratorConcept, L)
03:   If GeneratorConcept.Intent = intent
04:     Return GeneratorConcept
05:   End If
06:   GeneratorParents := GetParents(GeneratorConcept, L)
07:   NewParents =  $\emptyset$ 
08:   For each Candidate in GeneratorParents
09:     If Candidate.Intent  $\not\subseteq$  intent
10:       Candidate := AddIntent(Candidate.Intent  $\cap$  intent,
                               Candidate, L)
11:     End If
12:     addParent := true
13:     For each Parent in NewParents
14:       If Candidate.Intent  $\subseteq$  Parent.Intent
15:         addParent := false
16:       Exit For
17:       Else If Parent.Intent  $\subseteq$  Candidate.Intent
18:         Remove Parent from NewParents
19:       End If
20:     End For
21:     If addParent
22:       Add Candidate to NewParents
23:     End If
24:   End For
25:   NewConcept := (GeneratorConcept.Extent, intent)
26:   L := L  $\cup$  {NewConcept}
27:   For each Parent in NewParents
28:     RemoveLink(Parent, GeneratorConcept, L)
29:     SetLink(Parent, NewConcept, L)
30:   End For
31:   SetLink(NewConcept, GeneratorConcept, L)
32:   Return NewConcept

```

The parameters of the function *AddIntent(intent, GeneratorConcept, L)* are the *intent* of a new concept to be placed into the concept lattice *L* and a pre-computed *GeneratorConcept*, such that *intent* is a subset of the intent of *GeneratorConcept*. *AddIntent* returns a concept whose intent corresponds to *intent* – a new concept will be created if there was no such concept before or an existing one will be returned otherwise.

First, the algorithm finds the most general concept whose intent is a superset of *intent* (line 02) and assigns it to *GeneratorConcept*. If the intent of this concept is equal to *intent*, then the desired concept is already in the lattice and the algorithm terminates (line 04). Otherwise, *GeneratorConcept* is the canonical generator of the new concept, which has to be created and linked to other concepts in the diagram graph.

To find the parents of the new concept in the diagram graph, we examine all parents of *GeneratorConcept* (lines 08-24). If the intent of such a parent, called *Candidate*, is a subset of *intent*, then *Candidate* is modified. Otherwise, a recursive call to *AddIntent* ensures that the lattice contains a concept whose intent is equal to the intersection of *intent* and the intent of *Candidate*. This concept is assigned to *Candidate* (line 10). Then, *Candidate* is added to the (initially empty) *NewParents* list if it is minimal among its current elements (that is, has a maximal intent). At the same time, if some concept in *NewParents* is more general than *Candidate*, this concept is removed from the list (line 18). Thus, the *NewParents* list always contains incomparable (w.r.t. being more general) concepts. Moreover, in the end, it contains precisely the parents of *NewConcept* that is to be inserted (line 25). Proposition 3 below provides basis for the outlined procedure and Corollary 1 shows a way to optimize it (there is no need to test modified candidates for being minimal):

Proposition 3. (F', F) , then the parents of (F', F) in L_{i+1} are exactly the least general concepts from the set $\{(D', D) \mid D = F \cap H \text{ for some parent } (H', H) \text{ of } (B', B) \text{ in } L_i\}$.

Corollary 1. If (B', B) is the canonical generator of (F', F) , then every modified parent of (B', B) in L_i is a parent of (F', F) in L_{i+1} .

Thus, having processed the parents of *GeneratorConcept*, we create *NewConcept* with intent equal to *intent* and link it to concepts in the *NewParents* list, taking care to remove existing links between these concepts and *GeneratorConcept* (lines 27-30). Finally, *NewConcept* is set to be an upper neighbor of *GeneratorConcept* (line 31), and the *AddIntent* function returns *NewConcept*.

Note that the *AddIntent* function as described above does not update extents. Such an update is performed by the calling procedure constructing the lattice (shown below), but it can be easily integrated into *AddIntent* as well.

```

01: Procedure CreateLatticeIncrementally ( $G, M, I$ )
02:   BottomConcept :=  $(\emptyset, M)$ 
03:    $L := \{\textit{BottomConcept}\}$ 
04:   For each  $g$  in  $G$ 
05:     ObjectConcept = AddIntent( $g', \textit{BottomConcept}, L$ )
06:     Add  $g$  to the extent of ObjectConcept
       and all concepts above
07:   End For

```

After adding an object of the context to the lattice via a call to *AddIntent*, the procedure updates the extents of the concepts above and including the newly generated *ObjectConcept*.

```

01: Function GetMaximalConcept(intent, GeneratorConcept, L)
02:   parentIsMaximal := true
03:   While parentIsMaximal
04:     parentIsMaximal := false
05:     Parents := GetParents(GeneratorConcept, L)
06:     For each Parent in Parents
07:       If intent  $\subseteq$  Parent.Intent
08:         GeneratorConcept := Parent
09:         parentIsMaximal := true
10:       Exit For
11:     End If
12:   End For
13:   Return GeneratorConcept

```

In the *GetMaximalConcept* function, we use the diagram graph to find the most general concept, whose intent is a superset of the input *intent*. If this concept is not *GeneratorConcept*, then it is among its predecessors; therefore, it is sufficient to examine the predecessors of *GeneratorConcept*. If a more general concept is found (line 8), no further processing of *Parents* is required and this more general concept can be tested in the next iteration of the *While* loop.

The algorithms described above admit a number of optimizations, which are discussed in [12] but are not included here due to space limitations. For example, the number of set operations in the *GetMaximalConcept* function can be reduced by precomputing the number of attributes of g' that each concept in L has in common with g' since $|intent| = |Parent.Intent \cap g'|$ implies $intent \subseteq Parent.Intent$ (line 07). By first considering the candidate concepts whose intents yield the largest intersection with the new object intent, unnecessary traversal of the lattice can be avoided.

A worst-case time complexity bound of $O(|L||G|^3|M|)$ for the above version of the algorithm (i.e., the *CreateLatticeIncrementally* procedure) is derived in [13] (where $|L|$ is the number of concepts in the resulting lattice). A detailed discussion is not possible here due to space limitations, but the main argument is as follows. The complexity depends on the total number of invocations of the *AddIntent* function. The same intent can be passed as a parameter of *AddIntent* several times, but, clearly, the execution of *AddIntent* will go past line 05, i.e., past the call to *GetMaximalConcept*, at most once for every intent. Thus, for convenience, we may assume that *GetMaximalConcept* is called before the invocation of the *AddIntent* function, which, in its turn, is called only if the intent of the returned “maximal concept” is different from *intent*. In this setting, *AddIntent* would be called at most once for every intent of the lattice through the insertion of all objects. Since the length of the *GeneratorParents* list never exceeds $|G|$ and the complexity of *GetMaximalConcept* (as defined here) is bounded by $O(|G|^2|M|)$, the complexity of a single invocation of *AddIntent* (without a recursive call) can be estimated as $O(|G|^3|M|)$, which leads us to the total complexity of $O(|L||G|^3|M|)$ as stated above.

By introducing the optimizations referred to, this bound can be improved to $O(|L||G|^2 \cdot \max(|g'|))$ where $\max(|g'|)$ is the maximum number of attributes of any ob-

ject in the context [12]. For the purpose of direct comparison with other algorithms and since $|g'| < |M|$, we may replace $|g'|$ with $|M|$. A slightly less sharp complexity bound for the optimized algorithm is therefore $O(|L||G|^2|M|)$. The lowest available upper complexity bound of a lattice construction algorithm is that of Nourine [11] with $O((|G| + |M|)|G||L|)$ which is one factor better than the current estimate for the proposed algorithm. However, the experiments in Section 5 suggest that the latter is not as sharp as it might be, and that these bounds are not always good predictors of empirical performance.

4 An Example

Fig. 1. shows the relevant part of a concept lattice before the insertion of an object g with $g' = \{a, b, d, e, f, g, h\}$. Parts of the lattice that are not shown are indicated with lines ending in small solid circles – these concepts will not be considered by AddIntent due to the focused way it searches (Propositions 1–3). Concepts are named for reference purposes and only the concept intents are shown. Concepts shaded in grey are modified concepts whilst those in black are canonical generators. Non-canonical generators are filled in with diagonal lines whilst old concepts are not shaded.

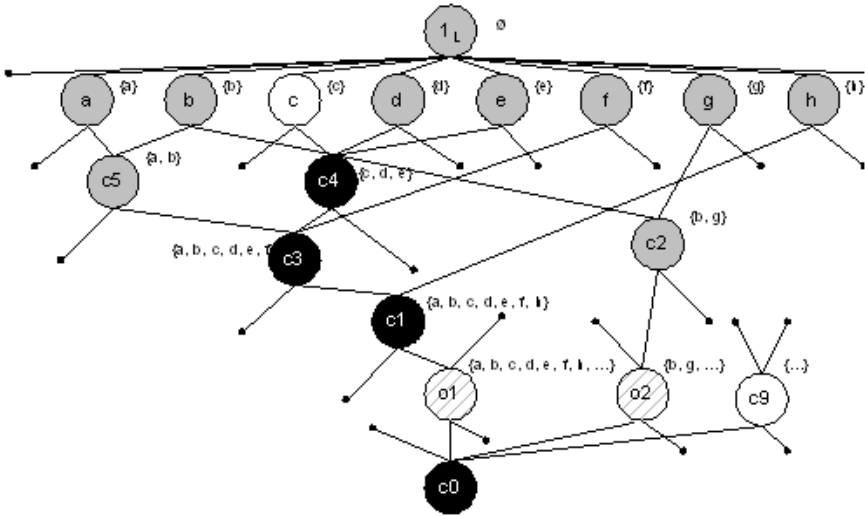


Fig. 1. Part of a concept lattice before inserting g with $g' = \{a, b, d, e, f, g, h\}$.

During the initial call $AddIntent(\{a, b, d, e, f, g, h\}, c_0, L)$, the concept c_0 is already maximal with respect to *intent* and it remains the *GeneratorConcept*. The object concept o_1 will be considered as the first *Candidate* (line 8). Since it is not modified, the intersection $\{a, b, d, e, f, h\}$ will be formed and a concept with such intent will be searched for or added by the recursive call $AddIntent(\{a, b, d, e, f, h\}, o_1, L)$ (line 10). After all subsequent recursive calls have unwound, the concept c_8 will be returned and added to *NewParents* before iterating through the rest of c_0 's parents. Fig. 2 contains a

table that traces important variables during all the recursive calls. The level of the recursion is indicated by the first two columns. Within a recursive call, several candidates can be considered; they are shown in the third column. The last column corresponds to the state of the *NewParents* list before the current *Candidate* is considered.

Intent	Generator (line2)	Candidate (line 8)	Candidate.Intent ∩ Intent	NewParents
{a, b, d, e, f, g, h}	c ₀	o ₁	{a, b, d, e, f, h}	∅
{a , b, d, e, f, h}	c ₁	c ₃	{a, b, d, e, f}	∅
{a, b, d, e, f}	c ₃	c ₅	{a, b}	∅
		c ₄	{d, e}	{c ₅ }
{d, e}	c ₄	c	∅	∅
∅	1 _L	–	–	–
{d, e}	c ₄	d	{d}	{1 _L }
		e	{e}	{d, e}
Create c ₆ : intent {d, e}; upper neighbors: d, e; lower neighbors: c ₄				
{a, b, d, e, f}	c ₃	f	{f}	{c ₅ , c ₆ , f}
Create c ₇ : intent {a, b, d, e, f}; upper neighbors: c ₅ , c ₆ , f; lower neighbors: c ₃				
{a, b, d, e, f, h}	c ₁	h	{h}	{c ₇ }
Create c ₈ : intent {a, b, d, e, f, h}; upper neighbors: c ₇ , h; lower neighbors: c ₁				
{a, b, d, e, f, g, h}	c ₀	o ₂	{b, g}	{c ₈ }
{b, g}	c ₂	–	–	–
Create c ₁₀ (g): intent {a, b, d, e, f, g, h}; upper neighbors: c ₈ , c ₂ ; lower neighbors: c ₀				

Fig. 2. Trace of important variables during the insertion of g into the lattice in Fig. 1.

Concept c_0 has additional upper neighbors (e.g., c_9) not included in the trace above. Since these have no attributes in their intents in common with g' , there will be exactly one recursive call of *AddIntent* for every such neighbor, and every such call will immediately get us to the top concept (line 2). The resulting lattice is shown in Fig. 3.

5 Experimental Comparison

In [8], a large number of lattice construction algorithms have been studied both theoretically and experimentally. We used the same implementations of the algorithms, and the results reported here may be seen as an extension of the work. Our experimental comparison shows that *AddIntent* performs quite well in constructing lattices from many types of datasets as compared to other incremental and batch algorithms and is often the best performer.

The charts below show the running time for *AddIntent*, as well as for several other algorithms (only the most efficient or the most popular ones were included): **Norris** [6], **NextClosure** [4], a version of **Bordat** [3] from [8], **Godin** [5], and **Nourine** [11]. Some of the algorithms were modified, since, in the original version, they produced only the concept set without the diagram graph. These modifications are described in [8].

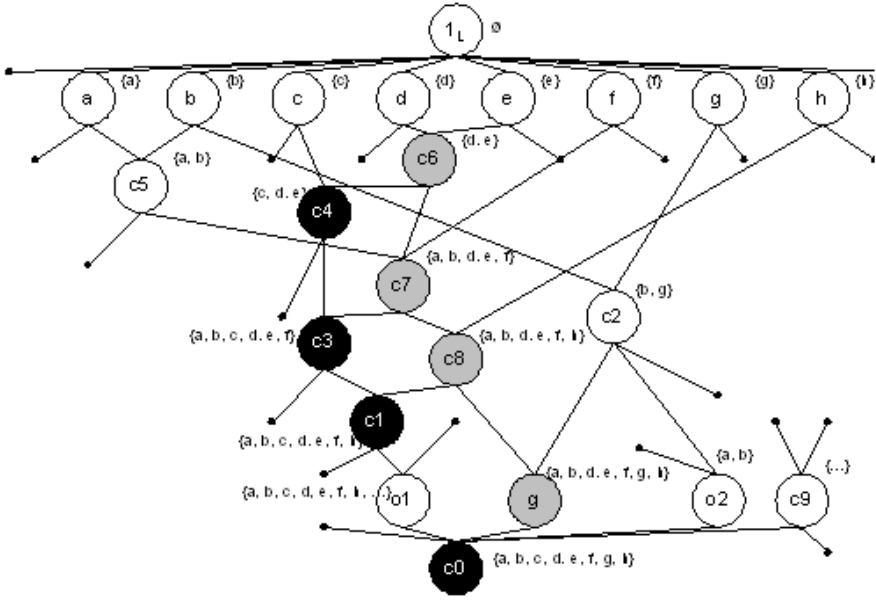


Fig. 3. The relevant part of the lattice resulting from inserting g with $g' = \{a, b, d, e, f, g, h\}$ into the lattice in Fig. 1.

Tests were performed on a Pentium 4 - 2GHz with 1 Gigabyte RAM using both randomly generated contexts and real datasets. In all random contexts, the total number of attributes is 100, and the number of objects varies, which leads to lattices of different sizes. The number in the name of a random context denotes its density (in percent), which, in this case, is the number of attributes per object $|g'|$ (all objects of such a context have the same number of attributes). In the following charts, we plot the time against the lattice size. Points on the curves denote an increment in the size of the object set.

For the sake of consistency, the AddIntent implementation used in the comparison is that of the algorithm described in Section 3 and does not include the optimizations referred to at the end of Section 3.

On random contexts with very low density (4%), the Bordat algorithm performs the best with AddIntent coming second. According to our other experiments, the algorithm proposed in [14] is even more suitable for very sparse contexts.

On random contexts with relatively modest density (25%), the AddIntent algorithm performs the best with Godin and Bordat coming next.

On random contexts with relatively high density (50%), AddIntent is still the best performer with Norris second and Nourine a close third.

In artificial contexts that create Boolean lattices, AddIntent is again among the best algorithms and, in fact, the best of the six algorithms presented here with Norris a close runner-up. In fact, our results indicate that the fastest algorithm on such contexts is [15], but the performance of AddIntent is comparable to the performance of [15]. However,

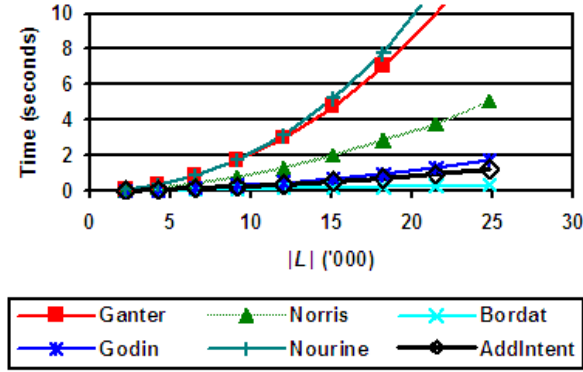


Fig. 4. Results for the Rnd-4 dataset: $|M| = 100$, $|g'| = 4$, $|G|$ varies from 100 to 900, the incremental step being 100 objects.

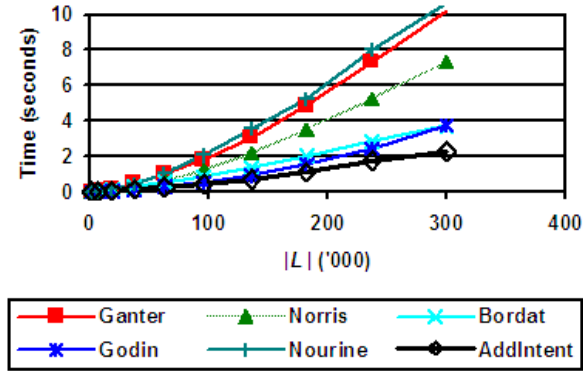


Fig. 5. Results for the Rnd-25 dataset: $|M| = 100$, $|g'| = 25$, $|G|$ varies from 10 to 100, the incremental step being 10 objects.

these contexts are primarily of theoretical interest, as they constitute the worst case with an exponential number of concepts and, thus, exemplify the worst-case complexity.

Fig 8. shows the results of several tests performed using real datasets from the UCI repository [16]. SPECT (Single Proton Emission Computed Tomography) is a real dataset that contains 267 objects and 23 attributes, generating a lattice with 21550 concepts. The remaining datasets (Breast Cancer¹, Wisconsin Breast Cancer, and Solar Flare databases) are given in the form of many-valued tables and the QuDA program [17, 18] was used to scale them into one-valued contexts.

It is interesting to note that the performance gap between AddIntent and the nearest best performers is even bigger than with artificial contexts (note that a logarithmic scale is used in this particular chart). Furthermore, even algorithms building only the concept

¹ This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

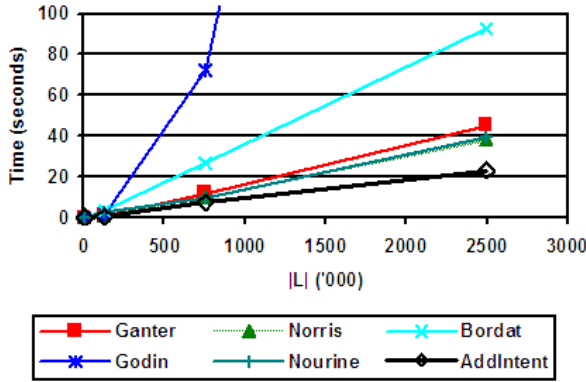


Fig. 6. Results for the Rnd-50 dataset: $|M| = 100$, $|g'| = 50$, $|G|$ varies from 10 to 40, the incremental step being 10 objects.

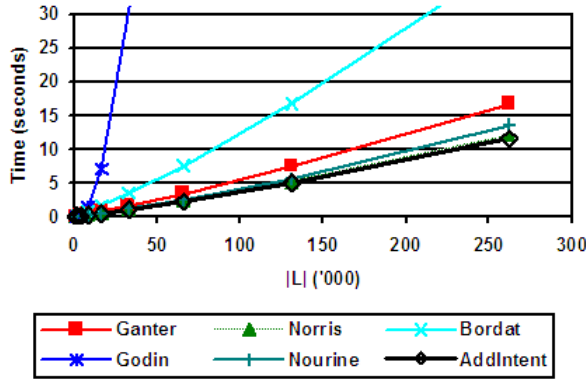


Fig. 7. Results for contexts of the form (G, G, \neq) , which give rise to Boolean lattices.

set were considerably slower than AddIntent on these datasets. Thus, on real databases, AddIntent seems to be the fastest algorithm among all available algorithms generating the concept set with- or without the diagram graph. Further research is however necessary to give more substance to this claim.

6 Related and Further Work

AddIntent was first implemented in 1996 in the context of so-called compressed pseudolattices [12, 19] and subsequently refined to the version given here. To the best of our knowledge, the most closely related lattice construction algorithm among those ever published is the recent one from [20] (designated “Algorithm 5” in the text). It uses a similar methodology based on Propositions 1 and 2 to avoid the consideration of old concepts and non-canonical generators in the lattice in a bottom-up search for canonical generators and modified concepts. This algorithm relies on a two-phased iterative ap-

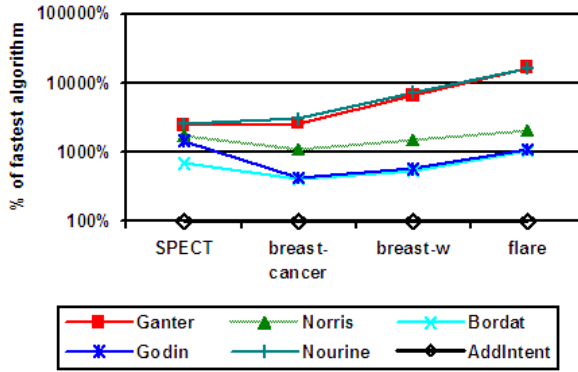


Fig. 8. Results for real datasets. The time spent by AddIntent is assumed to be 100%

proach to first discover canonical generators and in the second phase update the lattice. AddIntent combines these two phases. In addition, the algorithm in [20] uses a stack and *tries* as utility data structures facilitating lookup of concepts (as opposed to AddIntent’s use of recursion and the diagram graph itself). Direct comparison of this algorithm and AddIntent (and its optimizations) has not yet been conducted and is a topic of further research. Since both algorithms follow the same main strategy, one can expect that their behavior is also similar, in particular, as compared to the behavior of other algorithms. A deeper comparison may reveal potential trade-offs between specific issues in which the algorithms differ and may suggest further implementation improvements. We would like to thank the reviewers of this paper for pointing out the resemblance between our approach and the one in [20].

7 Conclusions

In [8], it has been shown that the choice of an algorithm should be based on the properties of the context such as its size and density, which are indeed good predictors for randomly generated contexts. From the results presented above, one can see that various algorithms perform differently across various context densities, whereas the performance of AddIntent is always at least acceptable, if not the best. In cases when this algorithm is not the fastest one (for example, on sparse contexts where it is inferior to Bordat), its speed is comparable with that of the best performer. In other situations, it is often superior to other algorithms. Therefore, AddIntent is a good candidate for a universal algorithm generating the diagram graph.

Certainly, the construction of the diagram graph requires much computational effort; hence, algorithms generating only the concept set, are in general, faster than AddIntent. However, as already mentioned, the situation with real datasets is apparently different. In real data, unlike in randomly generated contexts, we can expect specific patterns, and a good algorithm should be able to take advantage of them. If further experiments confirm the efficiency of AddIntent over other algorithms on real datasets,

this can be a step forward in lattice construction, since real datasets are the ones that matter after all.

A wider study is needed to understand how dependencies in real data can be exploited to get better performance. The proposed workshop on FCA algorithms [21] may provide the necessary infrastructure for this study.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag, Berlin Heidelberg New York (1999)
2. Wille, R.: Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. In Rival, I., ed.: Ordered Sets. Reidel, Dordrecht–Boston (1982) 445–470
3. Bordat, J.: Calcul pratique du treillis de Galois d’une correspondance. *Math. Sci. Hum.* **96** (1986) 31–47
4. Ganter, B.: Two Basic Algorithms in Concept Analysis. FB4-Preprint No. 831, TH Darmstadt (1984)
5. Godin, R., Missaoui, R., Alaoui H.: Incremental Concept Formation Algorithms Based on Galois Lattices. *Computation Intelligence* **11** (1995) 246–267
6. Norris, E.: An Algorithm for Computing the Maximal Rectangles in a Binary Relation. *Revue Roumaine de Mathématiques Pures et Appliquées* **23** (1978) 243–250
7. Valtchev, P., Missaoui, R.: Building Concept (Galois) Lattices from Parts: Generalizing the Incremental Methods. In: Lecture Notes in Artificial Intelligence. Volume 2120., Berlin Heidelberg New York, Springer-Verlag (2001) 290–303
8. Kuznetsov, S., Obiedkov, S.: Comparing Performance of Algorithms for Generating Concept Lattices. *J. Experimental and Theoretical Artificial Intelligence* **14** (2002) 189–216
9. Van Der Merwe, F., Kourie, D.: AddAtom: an Incremental Algorithm for Constructing Concept- and Concept Sublattices. Technical report of the Department of Computer Science, University of Pretoria, South Africa (2002)
10. Carpineto, C., Romano, G.: A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval. *Machine Learning* **24** (1996) 95–122
11. Nourine, L., Raynaud, O.: A fast algorithm for building lattices. *Information Processing Letters* **71** (1999) 199–204
12. Van Der Merwe, F.: Constructing Concept Lattices and Compressed Pseudo-Lattices. M.Sc. dissertation, University of Pretoria, South Africa (2003)
13. Obiedkov, S.: Algorithms and Methods of Lattice Theory and Their Application in Machine Learning. PhD thesis, Russian State University for the Humanities (2003)
14. Ferré, S.: Incremental Concept Formation Made More Efficient by the Use of Associative Concepts. INRIA Research Report no 4569 (2002)
15. Valtchev, P., Missaoui, R., Lebrun, P.: A Partition-Based Approach towards Building Galois (Concept) Lattices. Rapport de recherche no. 2000-08, Département d’Informatique, UQAM, Montréal, Canada (2000)
16. Blake, C., Merz, C.: UCI Repository of Machine Learning Databases. [<http://www.ics.uci.edu/~mlearn/MLRepository.html>], Irvine, CA: University of California, Department of Information and Computer Science (1998)
17. Grigoriev, P., Yevtushenko, S.: Elements of an Agile Discovery Environment. In: Lecture Notes in Artificial Intelligence. Volume 2843., Berlin Heidelberg New York, Springer-Verlag (2003) 309–316
18. Grigoriev, P., Yevtushenko, S., Grieser, G.: QuDA, a Data Miner’s Discovery Environment. Technical Report AIDA-03-06, TU Darmstadt [<http://www.intellektik.informatik.tu-darmstadt.de/~peter/QuDA.pdf>] (2003)

19. Van Der Merwe, F., Kourie, D.: Compressed Pseudo-Lattices. *J. Expt. Theor. Artif. Intell.* **14** (2002) 229–254
20. Valtchev, P., Missaoui, R., Godin, R., Meridji, M.: Generating Frequent Itemsets Incrementally: Two Novel Approaches Based on Galois Lattice Theory. *J. Expt. Theor. Artif. Intell.* **14** (2002) 115–142
21. Mailing list on FCA-algorithms:
<http://www.aifb.uni-karlsruhe.de/mailman/listinfo/fca-algo>.
Internet site of the Workshop on Algorithms of Formal Concept Analysis:
<http://kvo.itee.uq.edu.au/twiki/bin/view/Tockit/AlgoWorkshop>.

QuDA: Applying Formal Concept Analysis in a Data Mining Environment

Peter A. Grigoriev and Serhiy A. Yevtushenko

Technische Universität Darmstadt (TUD), Germany
{peter,sergey}@intellektik.informatik.tu-darmstadt.de

Abstract. This contribution contains a report on using FCA technologies in a data mining environment QuDA. We also show how “scaling” capabilities of QuDA can be used to transform real-world datasets into formal contexts.

Introduction

QuDA [GYG03] is a software environment for those who want to utilize support of machine learning in discovery tasks. This software includes various machine learning techniques, such as association rule mining, decision tree induction, JSM-reasoning, Bayesian learning, and interesting subgroup discovery. It provides also a number of error estimation and model selection tools and offers several preprocessing and postprocessing utilities. QuDA is a freeware available for download at <http://ki-www2.intellektik.informatik.tu-darmstadt.de/~jsm/QDA>.

In this paper we provide a report on using techniques of the Formal Concept Analysis (FCA) in QuDA.

1 Conceptual Scaling in QuDA

Conceptual scaling [GW99] is a process of transforming multi-valued contexts into unary-valued ones. A (conceptual) *scale* is a formal context that determines this procedure for a certain multi-valued attribute.

If values of a multi-valued attribute are partially ordered, the corresponding scale can be constructed automatically. Using this fact, to allow conceptual scaling QuDA employs a non-standard notion of an attribute type, which is unusual for a data analysis software. The type of an attribute in QuDA is intrinsically a *procedure*, that defines the generalization (meet) of any two of its values and thus imposes a partial order¹.

In fact, QuDA supports an infinite number of built-in attribute types, as some of the generalization procedures can be parameterized. One can also define custom attribute types by means of the *hierarchy editor* of QuDA (Fig. 1).

The most commonly used attribute types and the corresponding generalization (meet) operations are listed below:

¹ On using meet operator for handling structural data with FCA consider, e.g., the framework of pattern structures [GK01].

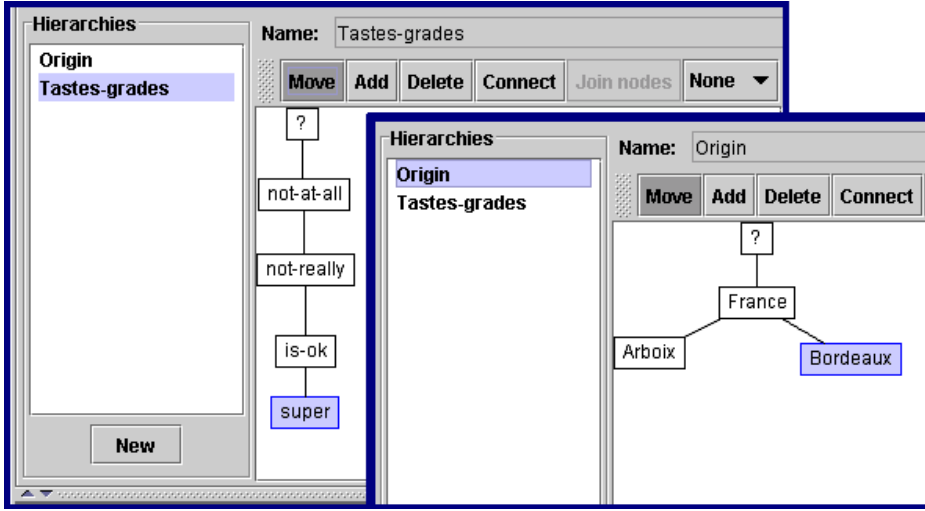


Fig. 1. QuDA's hierarchy editor is a means of introducing custom attribute types

1. *Nominal*. Acceptable are any values. For two values A and B , $meet(A, B) = A$, iff $A = B$, and ? ² otherwise.
2. *Quasi-unary*. Acceptable are values “1” and “?”. The meet operator is defined the same way as for the nominal type.
3. *Interval*. Acceptable values are real numbers and numeric intervals. The meet operator of two values is defined as a minimal (with respect to inclusion) interval, containing both values³:

$$meet([a, b], [c, d]) = [\min(a, c), \max(b, d)]$$

4. *Ordinal: Minimum*. Acceptable values are real numbers. For two values a and b , $meet(a, b) = \min(a, b)$.
5. *Ordinal: Maximum*. Acceptable values are real numbers. For two values a and b , $meet(a, b) = \max(a, b)$.
6. *Set of maximal common substrings*. Acceptable values are strings over a fixed alphabet Σ and sets of strings. Let $a, b \in \Sigma^*$. By $a \sqsubseteq b$ we denote a fact that a is a substring of b , i.e. $\exists c, d \in \Sigma^* : b = cad$. Let A be a set of strings. By $MS(A)$ we denote the set of all maximal (w.r.t. ' \sqsubseteq ') elements of A : $MS(A) = \{x \mid \forall y \in A : (x \sqsubseteq y) \rightarrow (x = y)\}$. For two values A and B ⁴,

$$meet(A, B) = MS(\{x \mid \exists a \in A, b \in B : (x \sqsubseteq a) \wedge (x \sqsubseteq b)\})$$

² ? stands for the top element of the lattice. The reason why the symbol ? is used is because it often denotes missing values in data.

³ A numeric value a is treated as an interval $[a, a]$.

⁴ A string value s is treated as a set $\{s\}$.

Along with those listed above, QuDA supports a number of other attribute types. Note, that if a user cannot find an appropriate “built-in” attribute type, he or she can define one of his or her own using the hierarchy editor (see Fig. 1).

A similar notion of an attribute type is used in the *Cernato* data analysis tool from NaviCon GMBH [Nav]. However, in this software the set of supported attribute types is more limited, custom hierarchies are not supported, and for the numerical attributes users are requested to specify the scaling thresholds.

To summarize, we believe that the conceptual scaling capabilities of QuDA implemented through its attribute types will be of use for the FCA practitioners dealing with real-world datasets.

2 Data Visualization with Concept Lattices

QuDA provides several means of getting its users familiar with their data. Line diagrams are one of those means, and, in some cases, a very powerful one. The use of line diagrams for data visualization has been thoroughly studied for a couple of decades now [GW99].

The line diagram, shown in Fig. 2, displays the distribution of objects (in this case e-mail messages) over two attributes:

- ALL_NATURAL - this attribute is true, if the e-mail contains a phrase “all natural” or the like,
- target - this attribute is true, if the e-mail is classified by the user as a *spam*.

As one can read from the line diagram in Fig. 2, all the e-mails that contain the phrase “all natural” or the like (circa 1% of all the e-mails in this dataset) are spam.

3 Mining Classification Rules with Concept Lattices

Let D be a universe of instances, each of which described through a data vector $X = (x^{(1)}, \dots, x^{(k)})$. Let f be a (target) function that maps elements of D into one of n classes. The task of automatic classification consists in building a *classifier*, i.e. an approximation for f , given a (limited) set of training examples, i.e. pairs $\langle a, f(a) \rangle$, where a is some element of D . For an introduction to this topic consider [Mit97]. The machine learning algorithms that induce classifiers are often referred to as *inducers*. A number of different inducers that build different types of classifiers are implemented in QuDA. One of them is based on the JSM-reasoning [Fin91]. The classifiers built by this inducer are expressible as *classification rulesets*.

A classification ruleset is a set of rules of the form: $P_1, P_2, \dots, P_n \rightarrow V$, where P_1, P_2, \dots, P_n are certain criteria and V is an expected value of the target function if those criteria hold. Fig. 3 demonstrates a classification ruleset built with JSM-reasoning. The instances this ruleset applies to are e-mail messages. The target function is “spam e-mail”. For example, the rule 2 reads as *If an e-mail contains a phrase like “click below”, then it is spam.*

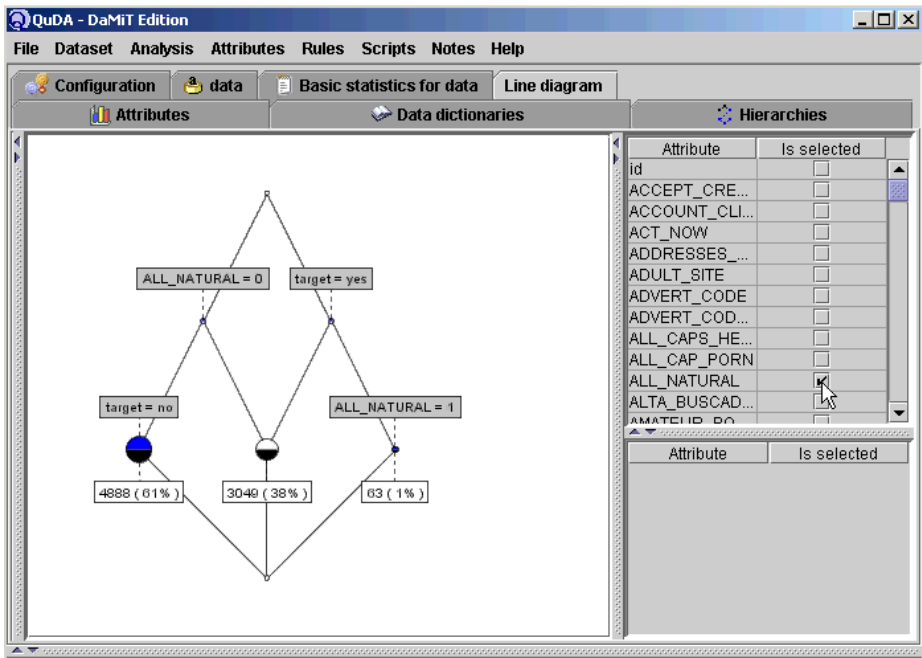


Fig. 2. Line diagram for the concept lattice of e-mail messages

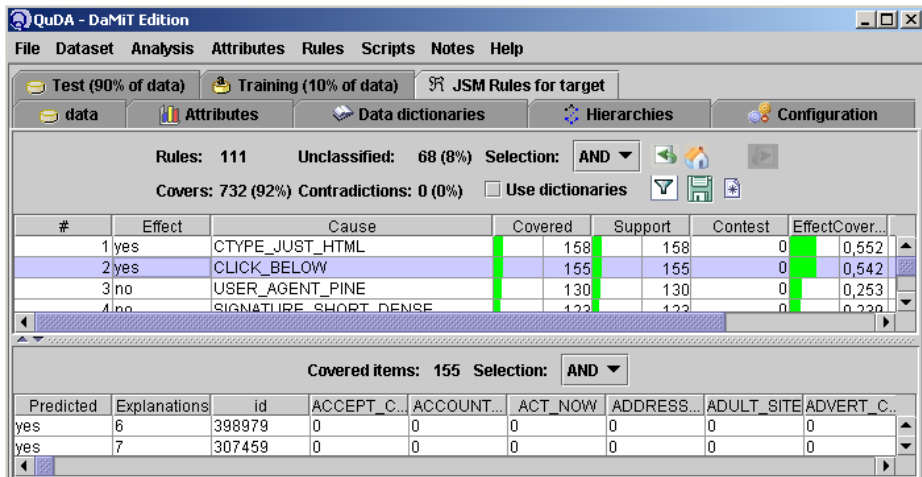


Fig. 3. A classification ruleset built with JSM-reasoning

One of the useful properties of classification rulesets is that they are easy to interpret, navigate, and “tune”. Another useful property is that classification rulesets easily allow background knowledge incorporation. QuDA exposes these properties in its convenient *rule navigator* (see Fig. 3). Our implementation

of the JSM-reasoning employs incremental bottom-up algorithms for building the set of formal concepts, similar to the one described in [Nor78]. For more information on JSM-reasoning in terms of FCA consider [GK00].

4 Mining Association Rules with Concept Lattices

Association rules are rules of the form $A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m$ where A_1, A_2, \dots, A_n , and B_1, B_2, \dots, B_m are unary predicates on instances (normally, value constraints for unary attributes). An association rule is described through its two characteristics, namely the *support*, i.e. the number of objects that satisfy all conditions in the premise and the conclusion of the rule, and the *confidence*, i.e. the ratio of the number of objects that satisfy conditions in the premise and the conclusion of the rule to the number of objects that satisfy conditions in the premise of the rule. The task of an association rule mining algorithm is to find all the association rules that satisfy the minimal *support* and minimal *confidence* criteria (see [AIS93] for details).

Quite recently it was shown that FCA can serve as a well-founded theoretical basis for this task [PBT98]. Several results, developed in FCA, most notably [Lux93], were transferred to the association rule mining task. The implementation for association rule mining in QuDA is partially based on these results.

QuDA allows to perform search for the association rules and then efficiently dig through the results by reducing the number of created rules with the help of Duquenne-Guigues-Luxemburger base (see, e.g. [Lux93]).

5 Mining Interesting Subgroups with Concept Lattices

The task of interesting subgroup discovery consists in searching for local patterns that identify subgroups of the explored population with some unusual, unexpected or deviated distribution of the target variable [Klo02b]. Subgroup descriptions should be interpretable and statistically significant. Subgroup discovery settings consist of a *description language*, which determines the space of subgroups, and *pattern classes* that describe the criteria for the subgroup interestingness.

The user of QuDA can specify for each attribute a separate hypothesis language by means of the scaling mechanism described in Section 1. Upon specifying the hypothesis language, the user can choose one of the several deviation detection patterns [Klo02a]. After performing search for interesting subgroups, user can navigate and filter out discovered patterns by means of the *subgroup navigator*.

6 Experiments on “Benchmark” Datasets

A number of practical comparisons of algorithms for finding the set of all formal concept in a formal context has been carried out over the past several years [Gue90, GMA95, KO01]. However, the practical comparisons we have mentioned

are mostly based on artificially-generated formal contexts. To our knowledge, the behavior of the algorithms on a wide variety of real-world datasets has never been explored.

Using scaling abilities of QuDA we transformed a number of real-world datasets from several different sources into formal contexts (see Table 1).

Table 1. Characteristics of the datasets used in comparison

Name	Source	Objects	Attributes	Fill Ratio	Max Attr. Per Obj.	Concepts
post-operative	UCI	90	26	0,34	9	2378
zoo	UCI	101	28	0,3	12	379
lymph	UCI	148	54	0,25	19	15504
spect.all	UCI	267	23	0,33	22	21550
breast-cancer	UCI	286	43	0,23	10	9918
dbdata0	Cristofo	298	88	0,07	21	2692
primary-tumor	UCI	339	45	0,15	13	3743
MortonRolphRacialStats2	Pissarro	487	61	0,19	22	32017
breast-w	UCI	699	91	0,11	10	9824
tic-tac-toe	UCI	958	29	0,34	10	59505
flare	UCI	1389	49	0,27	13	28742
kr-vs-kp	UCI	3196	42	0,27	19	101121
mushroom	UCI	8416	128	0,18	23	238710

Most of the formal contexts were obtained by scaling the datasets from the UCI machine learning repository [BM98]. The *spect.all* dataset was obtained by merging the training and the test sets of the *spect* dataset from the UCI machine learning repository. The *dbdata0* dataset was obtained from the ARMiner software distribution [ARM]. The *MortonRolphRacialStats2* dataset was obtained from the Pissarro software distribution [Pis].

We have performed comparison of the following five algorithms:

- Ganter – an implementation of the classical NextClosure [GW99] algorithm⁵;
- GRAIL⁶ [Yev00];
- Norris [Nor78];
- Norris-Godin [KO01] – a variant of the Norris algorithm that uses additionally a heuristic of the Godin algorithm [GMA95];
- Nourine-Raynaud [NR99]⁷.

All the algorithms were implemented in the JavaTM programming language. The comparison was performed on an Athlon 2400+ PC with 512 MB of RAM.

⁵ The worst-case computational complexity of our implementation is $O(m^2nH)$, where m is the number of attributes, n is the number of objects and H is the size of the concept lattice.

⁶ Note that for association rule mining and interesting subgroup discovery QuDA uses a variant of GRAIL that builds only the *iceberg lattices*. For details on iceberg lattices consider [STB⁺02].

⁷ This algorithm was the first one that achieved computational complexity $O(mnH)$.

We did not apply the variants of the Norris algorithm on the datasets with more than 900 objects, because these algorithms demonstrated a dramatic increase of experimental run-time with the growth of the number of objects. The implementation of the Nourine-Raynaud algorithm run out of memory when trying to compute the set of concepts for the *mushrooms* dataset.

The results of the comparison are presented in Table 2.

Table 2. Run times of the algorithms in milliseconds

Name	Ganter	Grail	Norris	Norris-Godin	Nourine
post-operative	80	100	971	1012	100
zoo	30	20	140	171	20
lymph	1673	1412	10916	11797	1412
spect.all	1171	1181	28881	30083	2444
breast-cancer	651	931	13709	14430	1822
dbdata0	1232	291	4516	4747	531
primary-tumor	1382	220	7020	7341	701
breast-w	7281	2764	30744	32546	7581
MortonRolphRacialStats2	10675	2744	94957	100304	13329
flare	20580	14861			52936
kr-vs-kp	53917	32567			333109
mushroom	1775203	218043			

7 Conclusions

In this contribution we provided a report on using FCA techniques in the data mining environment QuDA. We also showed how “scaling” capabilities of QuDA can be used to transform real-world datasets into formal contexts. Finally, we showed in experiments that the computational power of the modern computers along with advances in FCA algorithms allow building full concept lattices for a variety of the “benchmark” datasets. Thus, the application of FCA technologies in data mining tasks for the datasets, comparable in size and structure to the ones listed in the Table 2 is nowadays possible.

On the other hand, data mining applications often require scalability, i.e. an ability to handle huge amounts of data. In case scalability is required, a strategy of building important fragments of concept lattices, e.g. iceberg lattices, could be applied. Thus the development of algorithms like TITANIC [STB⁺02] or the modification of the GRAIL used in QuDA for association rule mining seems to be quite of interest for the data mining practitioners.

References

- AIS93. Rakesh Agrawal, Tomasz Imielinsky, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*, pages 207–216, 1993.
- ARM. ARMiner software. <http://www.cs.umb.edu/~laur/ARMiner/>.

- BM98. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- Fin91. V.K. Finn. Plausible reasoning in intelligent systems of JSM-type. *Itoigi Nauki i Tekhniki, ser. Informatika*, (15):54–101, 1991. In Russian.
- GK00. B. Ganter and S.O. Kuznetsov. Formalizing hypotheses with concepts. In G. Mineau and B. Ganter, editors, *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'00*, volume 1867 of *Lecture Notes in Artificial Intelligence*, pages 342–356. Springer-Verlag, 2000.
- GK01. B. Ganter and S.O. Kuznetsov. Pattern structures and their projections. In G. Stumme and H. Delugach, editors, *Proc. 9th Int. Conf. on Conceptual Structures, ICCS'01*, volume 2120 of *Lecture Notes in Artificial Intelligence*, pages 129–142. Springer-Verlag, 2001.
- GMA95. Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental concept formation algorithms based on galois lattices. *Computation Intelligence*, 11(2):246–267, 1995.
- Gue90. A. Guenoche. Construction du treillis de galois d’une relation binaire. *Math. Inf. Sci. Hum.*, 109:41–53, 1990.
- GW99. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical foundations*. Springer-Verlag, Berlin Heidelberg New-York, 1999.
- GYG03. Peter Grigoriev, Serhiy Yevtushenko, and Gunter Grieser. QuDA, a data miners’ discovery environment. Technical Report AIDA–03–06, FG Intellektik, FB Informatik, Technische Universität Darmstadt, September 2003. <http://www.intellektik.informatik.tu-darmstadt.de/~peter/QuDA.pdf>.
- Klo02a. Willi Kloesgen. Deviation analysis. In Willi Kloesgen and Jan M. Zytkow, editors, *Handbook of data mining and Knowledge Discovery in Databases*, pages 344–360. Oxford University Press, 2002.
- Klo02b. Willi Kloesgen. Subgroup patterns. In Willi Kloesgen and Jan M. Zytkow, editors, *Handbook of data mining and Knowledge Discovery in Databases*, pages 47–50. Oxford University Press, 2002.
- KO01. Sergei O. Kuznetsov and Sergei A. Obiedkov. Comparing performance of algorithms for generating concept lattices. In *Proc. of ICCS'01 - Int'l Workshop On Concept Lattices Based KDD*, pages 35–47, 2001.
- Lux93. Michael Luxenberger. *Implikationen, Abhängigkeiten und Galois Abbildungen*. PhD thesis, Technische Hochschule Darmstadt, 1993.
- Mit97. Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- Nav. Homepage of NaviCon. <http://www.navicon.de>.
- Nor78. E. M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumanie de Mathématiques Pures et Appliquées*, (23(2)):243–250, 1978.
- NR99. L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71:199–204, 1999.
- PBTL98. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Pruning closed itemset lattices for association rules. In *Proceedings of the BDA French Conference on Advanced Databases*, pages 177–196, 1998.
- Pis. Pissaro system – picturing interactively statistically sensible association rules reporting overviews. <http://stats.math.uni-augsburg.de/Pissarro/>.
- STB⁺02. Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with titanic. *Data Knowledge Engineering*, 42(2):189–222, 2002.
- Yev00. Serhiy A. Yevtushenko. The research and development of the algorithms for the concept lattices construction. Master’s thesis, National Technical University of Ukraine “KPI”, 2000. In Russian.

A Parallel Algorithm to Generate Formal Concepts for Large Data

Huaiguo Fu and Engelbert Mephu Nguifo

CRIL-CNRS FRE2499, Université d'Artois - IUT de Lens
Rue de l'université SP 16, 62307 Lens cedex, France
{fu,mephu}@cril.univ-artois.fr

Abstract. One of the most effective methods to deal with large data for data analysis and data mining is to develop parallel algorithm. Although Formal concept analysis is an effective tool for data analysis and knowledge discovery, it's very hard for concept lattice structures to face the complexity of very large data. So we propose a new parallel algorithm based on the NextClosure algorithm to generate formal concepts for large data.

1 Introduction

Parallel algorithms and parallel computer provide a very effective method to deal with very large data for data analysis and data mining [11]. The design of parallel algorithm for very large data analysis and processing becomes the necessity of many enterprises. In fact, many sequential algorithms can be improved by parallel algorithms [8].

Concept lattice is considered as an effective tool for data analysis, data mining, machine learning, information retrieval, etc. The generation of concepts and concept lattices is an essential task for formal concept analysis (FCA). Several algorithms were proposed to generate concepts or concept lattices of a context, for example: Bordat, Ganter (NextClosure), Chein, Norris, Godin and Nourine, etc [6]. Experimental comparisons of these algorithms show that NextClosure algorithm [4] is one of the best for large and dense data [6, 2]. But the problem is that it still takes very high time cost to deal with huge data. One way to avoid this problem is to design parallel lattice-based algorithms. In this paper, we propose a new parallel lattice-based algorithm **ParallelNextClosure**.

ParallelNextClosure is based on the decomposition of search space to generate concept. It's different from other existing decomposition algorithms that generate concept lattice using the approach of context decomposition [9] which could build many overlapping search sub-spaces. ParallelNextClosure uses a new method to decompose the search space by generating different non-overlapping search sub-spaces. It is also different from ParGal algorithm[7]. ParGal is based on Bordat's algorithm, there are many communications within the processors. Our algorithm is based on NextClosure algorithm. It can freely decompose the search space in different partitions if there exists at least one concept in each partition. Each

partition is independent and only shares the same source data (context) with other partitions. We can generate concepts independently for each partition using one processor. ParallelNextClosure algorithm shows good scalability, and can be easily used for parallel and distributed computing.

We consider the reader familiar with the basic notions of FCA such as context, Galois connection, closure operator, concept and concept lattice [1, 10, 5].

The rest of this paper is organized as follows: A new parallel algorithm ParallelNextClosure will be proposed in the next section. The algorithm analysis will be discussed in section 3. The paper ends with a short conclusion in the last section.

2 Parallel Lattice-Based Algorithm

2.1 NextClosure Algorithm

The principle of the NextClosure algorithm uses the characteristic vector, which represents arbitrary subsets A of M , to enumerate all concepts for context (G, M, R) . Given $A \subseteq M$, $M = \{a_1, a_2, \dots, a_i, \dots, a_{m-1}, a_m\}$, $A \rightarrow A''$ is the closure operator. The lexicographically smallest set is \emptyset'' . The NextClosure algorithm proved that if we know an arbitrary set A , the next concept¹ with respect to the lexicographical order is $A \oplus a_i$, where \oplus is defined by

$$A \oplus a_i = (A \cap (a_1, a_2, \dots, a_{i-1}) \cup \{a_i\})''$$

$A \subseteq M$ and $a_i \in M$, a_i being the largest element of M with $A < A \oplus a_i$ by lexicographical order.

In other words, for $a_i \in M \setminus A$, from the largest element to smaller one of $M \setminus A$, we calculate $A \oplus a_i$, until we find the first time $A < A \oplus a_i$, then $A \oplus a_i$ is the next concept.

The NextClosure algorithm can generate concepts more rapidly than other lattice algorithms for large data, but it still takes too much time to deal with very large data. So we improve it and propose a new algorithm to deal with very large data, our main idea is to use a parallel method to generate concepts.

2.2 ParallelNextClosure Algorithm

We propose a new algorithm which divides the search space into certain partitions. Each partition contains at least one concept. The processing of each partition is independent from another one. This is the main idea of our algorithm ParallelNextClosure in order to reduce the communication cost.

Analyzing all concepts of a context and their search space, we define the ordered context, and analyze its property in order to decompose the search space.

¹ It is the smallest one of all concepts that is larger than A .

Definition 1. A context (G, M, R) is called **ordered context**, if the attribute set $M = \{a_1, \dots, a_i, \dots, a_{m-1}, a_m\}$, the attributes with the same objects are merged as one attribute, and for all $i, j \leq m$: if the size of the extent of a_i is smaller than the size of the extent of a_j , then $i < j$.

In the cross table of such an ordered context we take the first column for an attribute with minimal extent size and the last column for an attribute with maximal extent size. An ordered context has the same concept lattice or concept lattice as the context, it doesn't change any characteristic of the context. The ordered context is very easy to be generated from a context. For Parallel-NextClosure algorithm, we don't generate a data file for ordered context, the ordered context is only stored in main memory. We can prove that this arrangement can raise efficiency of our algorithm.

Partitioning the Search Space for Concepts. In fact, a concept is a subset of M , so all subsets of M are elements of the search space. So the size of search space for enumeration of all concepts is 2^m . This search space can be considered as the fold of some subsets of M . For example, we consider that the search space is formed by *folds*, where $fold_i$ is all subsets of $\{a_i, \dots, a_{m-3}, a_{m-2}, a_{m-1}, a_m\}$ that include a_i . Each *fold* is a search sub-space. Here we define **folding search sub-space** in order to decompose the search space.

Definition 2. Let the attribute set M of a context (G, M, R) be $\{a_1, \dots, a_i, \dots, a_{m-1}, a_m\}$. Given $\forall a_i \in M$, let $S(a_i) = \{ \text{all subsets of the set } \{a_i, a_{i+1}, \dots, a_{m-1}, a_m\} \text{ which contain } a_i \}$, $S(a_i)$ is called **folding search sub-space (F3S)** of a_i .

For example, the folding search sub-space of attribute a_{m-3} is $\{a_{m-3}\}$, $\{a_{m-3}, a_m\}$, $\{a_{m-3}, a_{m-1}\}$, $\{a_{m-3}, a_{m-1}, a_m\}$, $\{a_{m-3}, a_{m-2}\}$, \dots , $\{a_{m-3}, a_{m-2}, a_{m-1}, a_m\}$, i.e. all subsets of $\{a_{m-3}, a_{m-2}, a_{m-1}, a_m\}$ that include a_{m-3} .

Proposition 1. Let (G, M, R) be a context, $M = \{a_1, \dots, a_i, \dots, a_{m-1}, a_m\}$, $\forall a_i \in M$, $n =$ the number of objects of attribute a_i , then the size of the folding search sub-space (for concept) of a_i is the minimum of 2^n and 2^{m-i} .

Proof: $\forall a_i \in M$, the set of $\{a_{i+1}, a_{i+2}, \dots, a_{m-2}, a_{m-1}, a_m\}$ has $m - i$ attributes. So according to the definition, the size of F3S of a_i is 2^{m-i} .

On the other hand, if the number of objects that verify attribute a_i is n , we have 2^n object subsets corresponding to a_i . So there are at most 2^n concepts that include attribute a_i .

Thus the size of the folding search sub-space of a_i is the minimum of 2^n and 2^{m-i} .

So we can order the context with the number of objects of each attribute. The smallest attribute is in the first column, and the biggest one is in the last column. In practice, this arrangement can remarkably allow to build an efficient algorithm for real data.

In the definition of ordered context, we need to merge the attributes with exactly the same objects as one item. The important reason for this is that we need to completely ensure that there are concepts in the folding search sub-space of an attribute. It's one important precondition of the following proposition.

Proposition 2. *For an ordered context, there are concepts in the folding search sub-space of an attribute.*

Proof: For an ordered context (G, M, R) , $\forall a_i \in M$ and $a_j \in M (1 \leq j < i)$, we have $\{a_i\}' \not\subseteq \{a_j\}'$, so $\{a_i\}''$ is in the folding search sub-space of attribute a_i , otherwise $\exists a_j (1 \leq j < i), \{a_i\}' \subseteq \{a_j\}'$.

This property of ordered context allows us to find partitions that include some search sub-space.

A Parallel Algorithm. We choose some attributes of ordered context to form an order set P . If the number of the elements of P is T , we have $a_{P_1} < a_{P_2} < \dots < a_{P_k} < \dots < a_{P_T}$. We denote $[a_{P_k}, a_{P_{k+1}}]$ as all F3S of attributes from a_{P_k} to $a_{P_{k+1}}$ for ordered context. From $\{a_{P_k}\}$ ($\{a_{P_k}\}$ is the first subset of $[a_{P_k}, a_{P_{k+1}}]$), we generate the next concepts until $\{a_{P_{k+1}}\}$, so that we can find all concepts between $[a_{P_k}, a_{P_{k+1}}]$.

All concepts (non-empty) of context are included in

$$\bigcup_{1 < k < T} [a_{P_k}, a_{P_{k+1}}] \cup [a_{P_T}]$$

We propose a new algorithm ParallelNextClosure which decomposes the search space and builds all concepts of each search sub-space. For each search sub-space we use the same method (NextClosure algorithm) to generate the closed sets. So we can generate all concepts of each search sub-space in parallel, as the search sub-spaces are independent.

ParallelNextClosure algorithm has two steps : determining the partitions and generating all concepts in each partition

In the first step of the algorithm, we can decide the number of partitions by a parameter $DP (0 < DP < 1)$ according to the size of data and our need. DP is used to determine the position of the beginning and the end of each partition.

We use all P_k to form the partitions $[a_{P_k}, a_{P_{k+1}}]$ and $[a_{P_T}]$, where $1 \leq k \leq T$. Here P_k means the position of an attribute of the ordered context, and we use it to represent the attribute a_{P_k} of the ordered context. a_{P_k} has a corresponding position in the initial context. When we search for the concepts, \emptyset isn't considered.

For each partition we compute the next concepts from $\{a_{P_k}\}$ to $\{a_{P_{k+1}}\}$. There is no relation between each partition. The partitions only share the same source data. We can deal with any partition independently.

Proposition 3. *Each partition is independent. In other words, we can find all concepts of one partition independently.*

Algorithm 1 Determining the partitions by the first processor

```

1: input a parameter  $DP$  ( $0 < DP < 1$ )
2: generate the ordered context (saving the order of attributes for ordered context in
   an array)
3: output the order of attributes of the ordered context
4:  $m$  = cardinal of the attribute set of the ordered context
5:  $min := m$ 
6:  $k := 0$ 
7: while ( $min >= 1$ ) do {determining partition}
8:    $k := k + 1$ 
9:    $P_k := min$ 
10:  output  $P_k$ 
11:   $min := int(min * DP)$ 
12: end while
13:  $T := k$  //  $T$  is the number of the partitions

```

Algorithm 2 Generating all concepts in each partition for one processor

```

1: input the order of attribute of ordered context
2: input  $P_k$  and  $P_{k+1}$  //input the partition
3:  $A \leftarrow \{a_{P_k}\}$ 
4:  $END \leftarrow a_{P_{k+1}}$ 
5:  $STOP := false$ 
6: while ( $!STOP$ ) do
7:    $A \leftarrow$  generate the next closure of  $A$  for the ordered context
8:   if  $END \in A$  when searching the next closure then
9:      $STOP := true$ 
10:  end if
11: end while

```

Proof: When we find the concepts of a partition, we only need to know the first subset and the last subset of this partition. And then we build next concepts from the first subset until the last subset, the computing is closed in this partition. So each partition is independent.

Proposition 4. All concepts of a context are included in the partitions. In other words, *ParallelNextClosure* algorithm generates the same concept lattice as *NextClosure* algorithm.

Proof: The partitions are obtained by partitioning the search space of all concepts. So all concepts of a context are included in the partitions.

Proposition 5. *ParallelNextClosure* algorithm always ends.

Proof: Each partition has a last subset, so it can end when generating all concepts in each partition with one processor. So *ParallelNextClosure* algorithm always ends.

We show below an example (using Figure 1) of *ParallelNextClosure* algorithm:

First, we need to generate the ordered context, the attribute set of the ordered context is: $a_5a_8a_6a_7a_4a_3a_2a_1$ (a_2 and a_9 are merged as a_2). And then, we can give a value to the parameter to determine the partitions, for example, $DP = 0.5$. We get 4 partitions: $[a_1, a_7]$, $[a_7, a_8]$, $[a_8, a_5]$ and $[a_5]$. At the end, we find all concepts in each partition. In fact, the search space of each partition is:

$$\begin{aligned} [a_1, a_7]: & a_1, a_2, a_2a_1, a_3, a_3a_1, a_3a_2, a_3a_2a_1, a_4, a_4a_1, \dots, a_4a_3a_2a_1 \\ [a_7, a_8]: & a_7a_1, a_7a_2, a_7a_2a_1, \dots, a_6, a_6a_1, \dots, a_6a_7a_4a_3a_2a_1 \\ [a_8, a_5]: & a_8, a_8a_1, a_8a_2, a_8a_2a_1, \dots, a_8a_6a_7a_4a_3a_2a_1 \\ [a_5]: & a_5, a_5a_1, a_5a_2, \dots, a_5a_8a_6a_7a_4a_3a_2a_1 \end{aligned}$$

3 Algorithm Analysis

In fact, ParallelNextClosure algorithm is suitable for real data. The worst case should be considered when we determine the partitions: the worst case appears when the sizes of G and M are equal to m , and each attribute is verified by $m - 1$ different objects, each object possesses $m - 1$ different attributes.

For the worst case, a different technique can be used to generate partitions in order to avoid a great unbalance of partitions in terms of the number of concepts. We can redecompose the search sub-space of an attribute into some partitions so that it's easy to deal with for each partition according to the number of concepts per partition. The aim is to decrease the complexity of each partition.

The problem of balance is always an important problem for parallel algorithm. The parameter $DP(0 < DP < 1)$ of ParallelNextClosure algorithm is used just to balance the partition size. The computing of each partition can be implemented on any free processor so that it can help to balance the processors.

4 Conclusion

In this paper, a new parallel lattice-based algorithm, ParallelNextClosure is proposed for creating the partitions and building concepts in each partition.

ParallelNextClosure uses a new method to decompose the search space. We have defined ordered context and folding search sub-space of an attribute. Furthermore, we have proved that there are concepts in the folding search sub-space of an attribute. Thus our method can freely decompose the search space in any partitions if there are concepts in each partition. Each partition is independent, so we can realize our parallel algorithm.

Experimentations with one processor show the following results [3]: on some data, our algorithm was 60 times faster than NextClosure. And we have generated all concepts for worst case data sets with 24, 25, 30, 35 and 50 attributes, which was very difficult to obtain with NextClosure on a 128MB RAM computer. The gain comes from lack of memory in the case of NextClosure.

The ongoing research is to study the problem of balance of all partitions. Further research will consist of implementation of this algorithm on a parallel platform and an analysis of the results.

	a ₁	a ₂	a ₃	a ₄	a ₅	a ₆	a ₇	a ₈	a ₉
1	×	×					×		×
2	×	×					×	×	×
3	×	×	×				×	×	×
4	×		×				×	×	
5	×	×		×		×			×
6	×	×	×	×		×			×
7	×		×	×	×				
8	×		×	×		×			

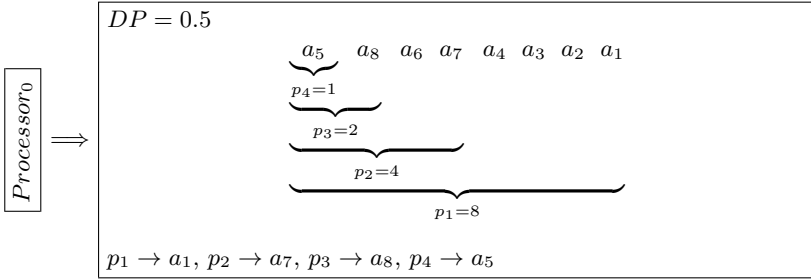
Context

\Rightarrow

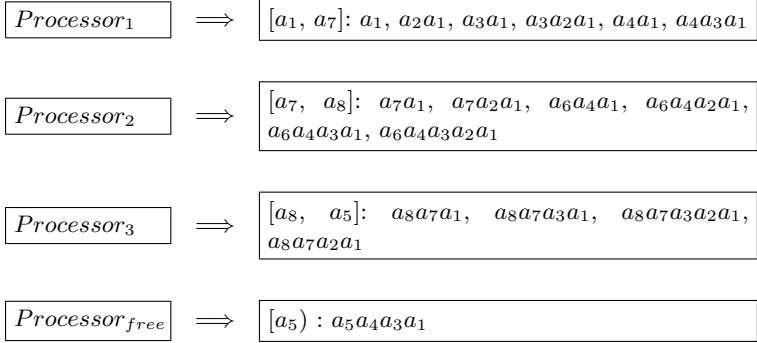
	a ₅	a ₈	a ₆	a ₇	a ₄	a ₃	a ₂	a ₁
1				×			×	×
2		×		×			×	×
3		×		×		×	×	×
4		×		×		×		×
5			×		×		×	×
6			×		×	×	×	×
7	×				×	×		×
8			×		×	×		×

Ordered context

Determining the partitions:



Generating Concepts:



Concept lattice:

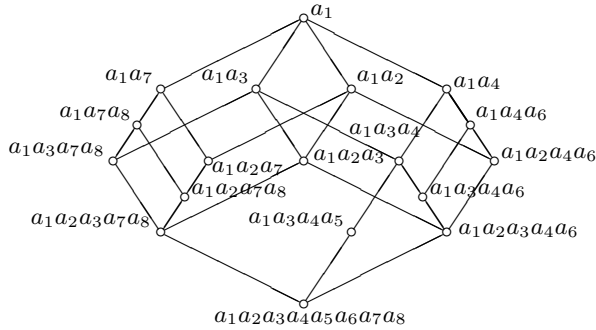


Fig. 1. An example with ParallelNextClosure algorithm using 3 processors

Acknowledgements

We are grateful to anonymous reviewers for helpful comments and to Jean Jacques Givry for english proofreading. This research benefits from the support of IUT de Lens, Université d'Artois, CNRS and the region Nord/Pas de calais.

References

1. G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, RI, 3rd edition, 1967.
2. Huaiguo Fu and Engelbert Mephu Nguifo. How well go lattice algorithms on currently used machine learning testbeds? In *ICFCA 2003, First International Conference on Formal Concept Analysis*, 2003.
3. Huaiguo Fu and Engelbert Mephu Nguifo. Partitioning large data to scale up lattice-based algorithm. In *Proceedings of ICTAI03*, Sacramento, CA, November 2003. IEEE Computer Press.
4. B. Ganter. Two basic algorithms in concept analysis. Technical Report 831, Technische Hochschule, Darmstadt, Germany, 1984. preprint.
5. B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations*. Springer, 1999.
6. S. Kuznetsov and S. Obiedkov. Comparing performance of algorithms for generating concept lattices. *JETAI Special Issue on Concept Lattice for KDD*, 14(2/3):189–216, 2002.
7. P. Njiwoua and E. Mephu Nguifo. A parallel algorithm to build Concept Lattice. In *Proceedings of the 4th Groningen International Information Technology Conference for Students*, pages 103–107, University of Groningen, The Netherlands, Février 1997.
8. M. Tchuente. *Parallel Computation on Regular Arrays*. Algorithms and Architectures for Advanced Scientific Computing. Manchester University Press, 1991.
9. Petko Valtchev and Rokia Missaoui. Building galois (concept) lattices from parts: Generalizing the incremental approach. In *Proceedings of the ICCS 2001, LNCS 2120*, pages 290–303. Springer Verlag, 2001.
10. R. Wille. Restructuring Lattice Theory. In Ivan Rival, editor, *Symposium on Ordered Sets*, pages 445–470. University of Calgary, Boston, 1982.
11. M.J. Zaki and C.-T. Ho. *Large-Scale Parallel Data Mining*. Springer, 2000.

Using Concept Lattices for Requirements Reconciliation

Debbie Richards

Department of Computing
Macquarie University, Sydney, Australia
`{richards}@ics.mq.edu.au`

Abstract. We have developed a requirements engineering process model based on viewpoint development. Our goal is to capture and reconcile a set of requirements in the form of use case descriptions so that a more complete and representative set of requirements will provide the basis for system design and development. Our technique converts the natural language sentences in each individual use case viewpoint into a formal context. We then apply Formal Concept Analysis to generate a concept lattice of the requirements. Our process model and tool allow viewpoints, use cases, sentences and phrases to be selected for comparison, tagging and reconciliation. The final product is a shared use case description that has been agreed upon by the project group. In this paper we provide an example developed by final year software engineering students who used the approach as a means to gain experience in the specification of use cases and the sometimes painful task of arriving at a shared view of the requirements before system design and development commenced. Our next step is evaluation with requirements engineers to develop a full-strength industrial product and process.

1 Introduction

We have developed a requirements engineering process model based on viewpoint development. Others (e.g. Darke and Shanks 1997, Easterbrook and Nuseibeh 1996, Finkelstein et al. 1989, Mullery 1979, Kotonya and Sommerville 1998) have also advocated viewpoint development as one possibility to handle some difficulties of requirements definition. What differentiates this work from others is the capture of requirements in natural language (which is far more acceptable and manageable to the average stakeholder than formal specification), automatic generation of a formal representation and the ability to identify and reconcile differences via a visualization of the original text. Formal Concept Analysis plays a key role in the generation and ordering of concepts from use case sentences. We call our approach RECOCASE as we offer a CASE (Computer Aided Software Engineering) tool (as shown in the screen dumps in this paper) to assist with viewpoint RECOnciliation. In this paper we follow an example which uses our requirements engineering process model. At the end we offer some results from initial evaluations, review of related literature and our future directions.

2 The RECOCASE Approach

The RECOCASE methodology includes five iterative phases: requirements capture; requirements translation; concept generation; concept comparison and reconciliation; and evaluation. In this section we follow an example that demonstrates the process and the techniques which we have novelly combined and applied to the problem of requirements reconciliation. The more technical and formal details regarding the use cases, Link Grammar, ExtrAns and RECOCASE-logic are given in (Richards and Böttger 2002). The sample problem we follow is the Automated Video System (AVS) which is being designed to allow customers to self check out videos, search for particular video titles, request new videos to be stocked by each store, and to make reservations for videos currently unavailable. The example is a real one being used by groups of final year software engineering students to develop a software product. We chose this problem as it is a new problem (such systems are not available in Australia) and thus the requirements are not completely understood but also not completely unlike other systems they are familiar with. By following the RECOCASE methodology students gain some understanding of the difficulties associated with acquiring requirements and managing a number of viewpoints. In the following subsections we take you through each phase. The example uses a reduced set of the actual sentences specified by students as to make the example easier to follow in the limited space available. A whole paper could be written on how to use one diagram to identify and reconcile differences to produce a shared use case.

2.1 Requirements Capture

While a number of approaches to requirements specification have been proposed, see a summary in Düwel (1999), we have chosen to adopt the increasingly popular technique of use case description. This first phase starts by a group, led by the group facilitator/leader (GF), brainstorming a set of use cases. Viewpoints for each use case and viewpoint representative are identified. Asynchronously, viewpoint representatives enter use case descriptions in natural language into the RECOCASE tool. Natural language can however be ambiguous and complex in syntax and semantics. A set of writing guidelines and controlled language is provided (Richards, Böttger and Aguilera 2001), but since users rarely follow guidelines, the tool can also detect and notify the viewpoint agent of problematic words, e.g. pronouns, modal verbs, and, or and negations.

2.2 Requirements Translation

The next step is to transform the sentences into a formal representation for later manipulation and comparison. To support comparison via a line diagram using FCA techniques we need to start with a formal context. Each use case forms a new formal context, with the sentences comprising the objects in the context table. Originally we considered a very naïve approach which used the words of the sentence as the crosstable attributes. However since sentences are not simply random strings of words and to allow reconstruction of the sentence by the

reader, we are able to use the flat logical forms (FLFs) output by ExtrAns using LinkGrammar as a semantic representation for the sentence which we transpose into word phrases using RECOCASE-logic. For example as shown in figure 1, LinkGrammar¹ provides the following syntactic structure and constituent tree for the sentence ‘The system updates the credit card on the member card.’

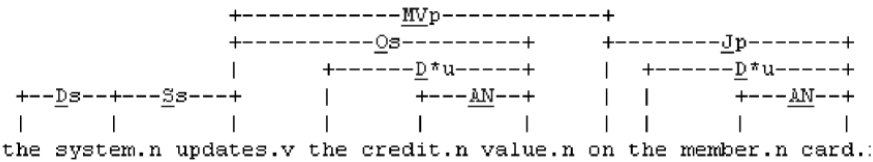


Fig. 1. FLF example

‘S’ connects the noun ‘system’ to the finite verb ‘update’, which is a transitive verb and again connected through ‘O’ with its object ‘value’. ‘AN’ connects the noun-modifier ‘credit’ with the noun ‘card’. ‘D’ connects determiners to nouns. ‘system’, ‘value’ and ‘card’ are in this way connected with the determiner ‘the’. ‘MV’ connects verbs and adjectives to modifying phrases which follow the verb like adverbs, prepositional phrases and subordinating conjunctions. In this example the verb ‘update’ is connected through ‘MV’ with the preposition ‘on’, which introduces a prepositional phrase. To the right the preposition ‘on’ is connected to its object ‘card’ through ‘J’.

The syntactic structure provided by LinkGrammar is used by ExtrAns to create FLFs as a semantic representation for the core meaning of each sentence.

The translation of FLFs into crosstable attributes follows an algorithm, known as RECOCASE-logic, which uses graph theory. The algorithm contains two main parts. The first part includes building a graph representing the FLFs of a sentence. The nodes of the graph represent the words of the FLF predicates and the edges represent the relations between them. The relation between words of the predicates is derived from the FLFs. At a coarse level the algorithm is looking for the main event, object or property which is defined by the predicate ‘holds’. If the FLF of a sentence does not contain the predicate ‘holds’, the root ‘A’ is defined through ‘if(A,B)’, ‘while(A,B)’ or ‘not(A)’ in this sequence. In a second step the graph is reduced to the crosstable attributes. Figure 2 shows the steps involved in creation of the graph for the sample sentence. Table 1 shows the crosstable produced by RECOCASE-logic for the following related sentences from three viewpoints: “The system updates the credit value on the member card”, “the system debits the credit of the card” and “the system deducts the price of the rental from the member card”.

If all sentences are translated using RECOCASE-logic a crosstable can be created as shown in Table 1. The words or word phrases form the set of attributes where no word or phrase should exist twice. All translated sentences form the set of objects.

¹ Using the parse sentence facility at <http://www.link.cs.cmu.edu/link/index.html>

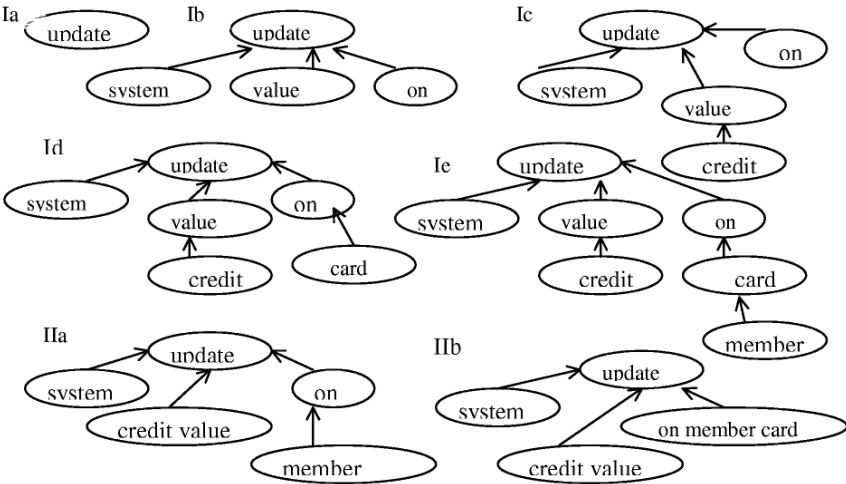


Fig. 2. Steps in the creation (Ia-Ie) and reduction (IIa-IIb) of the sentence “the system updates the credit value on the member card”

Table 1. Cross table for the sentences “The system updates the credit value on the member card”, “the system debits the credit of the card” and “the system deducts the price of the rental from the member card”

	(system)	(update)	(credit value)	(on member card)	(credit)	(debit)	(on card)	(deduct)	(price of rental)
AgentA	×	×	×	×		×			
AgentB	×			×	×	×			
AgentC	×			×			×	×	×

2.3 Concept Generation

Figure 3 shows the line diagram for the concept lattice generated from the formal context in Table 1 using standard FCA as described in (Ganter and Wille 1999). As can be seen in the crosstable and the line diagram the only shared attribute is 'system'. The line diagram provides structure and visual clarity of shared concepts. In the next phase we show how the line diagram is used to identify and manage conflicts.

2.4 Concept Comparison and Reconciliation

The GF creates the diagrams by selecting the sentences he/she wants to have included. Different strategies for combining use case descriptions can be used.

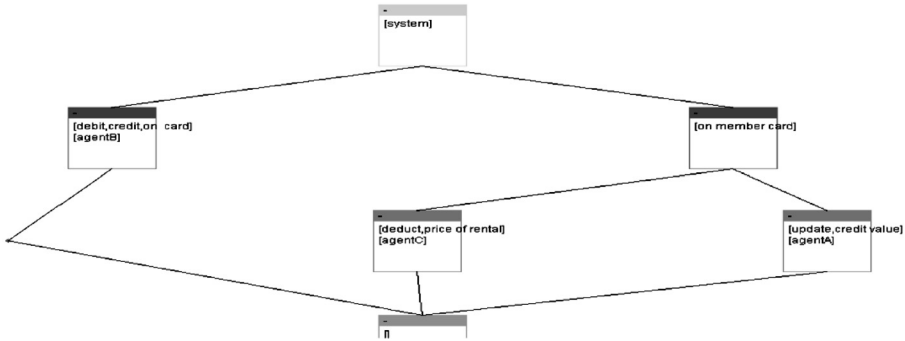


Fig. 3. Representation of the sentences from the context table in Table 1

If there are few sentences, maybe one or two diagrams constructed by selecting sentences based on system interaction and actors are adequate. However, if there are a larger number of sentences (≥ 25) the GF can choose to create diagrams based on logical units of sentences. The GF will then be able to have a mental overview of the logical flow in the use case. The group members and the GF can use the line diagram to compare viewpoints and detect conflicts. The nodes in the graph represent the concepts, and in our diagrams these are words and phrases from the use case description. The nodes belonging to the same sentence are connected with lines. Sentences containing identical words or synonyms will share nodes.

During the reconciliation process, the group and GF will look at the diagrams together to build a shared use case. Using the lattice we can determine if two concepts are in a state of consensus (same idea, same terms - identified by sharing the same node), correspondence (same idea, different terms - shares some nodes, synonym table needed to map terms), contrast (different idea, same terms - shares some nodes but appears incorrect) or conflict (different idea, different terms - no overlap of pathways). The two key strategies in handling reconciliation is to use a synonym table to map terms and the use of tags to indicate if a concept should be ignored, delayed or circumvented. Figure 4 shows the resulting diagram after some synonyms have been applied to Figure 1. See (Richards 2003) for greater explanation of the tags and our negotiation strategies.

This phase is essentially one involving group discussion under the guidance of the GF. We noted from our evaluation studies that in groups not using the RECOCASE methodology or tool, that participation in this phase was limited to a few interested individuals. It appears that when all participants are required to provide a viewpoint that they tend to engage with the problem and defend their own viewpoint.

2.5 Evaluation

We use graph theory on the lattices to evaluate if the different viewpoints have become more similar, or if a new round of negotiation is necessary. In the RE-

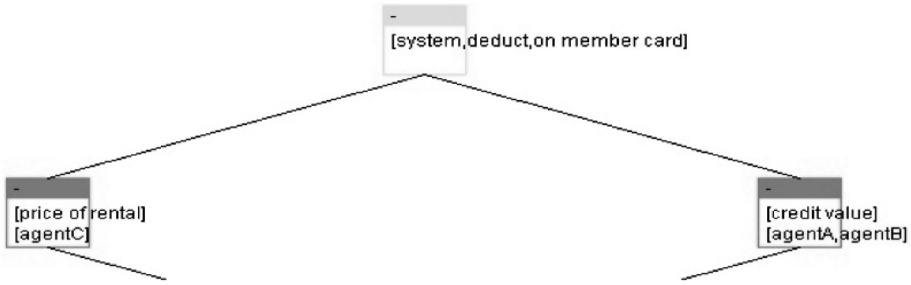


Fig. 4. Representation of the sentences in Figure 3 after a synonym table has been used to map deduct=update=debits, and 'on card' = 'on member card'. From this figure we can see we still need to map 'price of rental' = 'credit value' to bring all viewpoints into consensus

COCASE tool we can select viewpoints for comparison and see the percentage of similarity between descriptions. We can also see the percentage of similarity between each sentence in two different viewpoints. This information can be used to determine which viewpoints to try to reconcile first (e.g. reconcile most similar or most different first).

3 Discussion and Conclusion

We have conducted a number of different evaluations. Our first studies concentrated on the learnability and comprehensibility of the line diagram for reasoning about requirements. We found that after a 5 minute introduction, 58% of students were able to correctly interpret (no errors) and 80% more likely to give the correct answer to questions involving comparison of viewpoints using the diagram as opposed to the original text. A particularly significant difference was the time taken to answer questions using the diagram rather than text was up to (a number of tests were done) 9.9 times faster. Our recent studies evaluating the group process and tool revealed that each final and shared use case description was longer than the individual use case descriptions. All participants were satisfied with the process and the outcome and were able to create a shared use case description using the technique. The most striking difference was found in the attitudes and cohesion of the groups that used the process compared to the groups that did not.

The application of FCA for software engineering activities has been pursued by a number of researchers. A useful survey is provided in (Tilley et al. 2003). Whilst the application of FCA to the requirements phase has not been the primary area of activity targeted some work does exist. Andelfinger's (1997) thesis, reported in Tilley et al. 2003, concerned a question-answering and discussion tool using FCA which could be applied to the area of requirements capture. More directly relevant is the work of Düwel and Hesse (2000) who use FCA to identify potential classes and attributes from use case descriptions. Their work is different to ours in a number of ways. Firstly, in the formal context they build, the

use cases are the columns (attributes) and the rows (objects) are the potential classes or attributes. In our approach we develop one or more formal contexts for each use case, with objects being sentences and attributes being words and phrases, and the formal context usually contains at least two viewpoints of the complete or partial use case. Thus our focus is much more fine-grained and not restricted to certain parts of the sentence. Düwel and Hesse have also developed a tool to assist with entry of use case descriptions and development of concept lattices. As with our tool, they shield the user from the mathematical complexities of FCA. Also, as we do, they acknowledge that the decision process, which in their case concerns the choice of classes to use and how they relate to one another, “requires much detailed work which cannot completely be automated, including negotiations and a thorough discourse between domain experts and software developers”. However, the approach to creation of the formal context differs significantly from ours. Our approach takes in natural language sentences and automatically generates a formal context using natural language techniques. In their approach, the user is required to select words from a list or the description which become the set of objects. Given that our approach identifies nouns, objects and events (all possible candidates for classes), we envisage that our natural language techniques could be beneficially combined with the Düwel and Hesse approach. Indeed we believe that our ability to produce FLFs from the natural language descriptions offers a entry point and input solution for many other formal specification techniques.

As a result of the initial success of the approach with students, which has led to revision of the tool and fine tuning of the process substeps, we are seeking industrial collaboration to develop the approach further and incorporate it as an integral step of object oriented design using UML. As Düwel and Hesse (2000) stated “we find it quite an interesting observation that just a “non-OO” technique like use case analysis has become so popular as a starting point to OO analysis”, we also find it interesting that use case descriptions are textual and the rest of the UML process is visual. By offering an approach which visualizes the textual descriptions we provide a completely visual technique to system specification and design. Further, by capturing the requirements from multiple viewpoints we get a more complete and consistent set of requirements and a greater sense of ownership by the project group, thus providing a better basis for system design and increasing the likelihood of eventual system acceptance.

Acknowledgment

Many thanks to those who have contributed to this project: Oscar Aguilera, Kathrin Böttger, Anne Britt Fure, Rolf Schwitter and Diego Molla-Alloid. This project is funded by the Australian Research Council.

References

1. Darke, P., Shanks, G.: Managing user viewpoints in requirement definition. 8th Australasian Conference on Information Systems. 1995.
2. Düwel, S. (1999) Enhancing System Analysis by means of Formal Concept Analysis In Conference on Advanced Information Systems Engineering 6th Doctoral Consortium, Heidelberg, Germany.
3. Düwel, S and Hesse, W. (2000) Bridging the gap between Use Case Analysis and Class Structure Design by Formal Concept Analysis In J. Ebert and U. Frank (eds.) Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik. Proceedings "Modellierung 2000", Koblenz, 2000, Foelbach-Verlag, 27-40.
4. Easterbrook, S. and Nuseibeh, B. (1996) Using Viewpoints for Inconsistency Management BCSEEE Software Engineering Journal, January 1996, 31-43.
5. Finkelstein, A.C.W., Goedicke, M., Kramer, J. and Niskier, C. (1989) Viewpoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering In Proc. of the 2nd Meteor Workshop on Methods for Formal Specification, Springer Verlag, LNCS.
6. Ganter, B. and Wille, R. (1999) Formal Concept Analysis: Mathematical Foundations, Springer, Berlin.
7. Kotonya, G. and Sommerville, I. (1998) Requirements Engineering: Process and Techniques Chichester, UK, Wiley and Sons.
8. Mullery, G. P. (1979) CORE - a method for controlled requirements expression. In Proceedings of the 4th Int. Conf. on Software Engineering (ICSE-4), IEEE Computer Society Press, 126-135.
9. Richards, D (2003) Merging Individual Conceptual Models of Requirements, Spec. issue on Model-Based Requirements Engineering for the International Journal of Requirements Engineering.
10. Richards, D., Böttger, K and Aguilera, O. (2002) A Controlled Language to Assist Conversion of Use Case Descriptions, 15th Australian Joint Conference on Artificial Intelligence, Canberra, Australia, December 2nd - 6th, 2002.
11. Richards, D. and Böttger, K. (2002) Representing Requirements in Natural Language as Concept Lattices, In Proc. 22nd Annual Int. Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGES), (ES2002), Cambridge, UK, December 10-12, 2002
12. Tilley, T., R. Cole, P. Becker and P. Eklund, A Survey of Formal Concept Analysis Support for software engineering activities", in *Proceedings of the First International Conference on Formal Concept Analysis - ICFCA '03*, (eds) G. Stumme and G. Ganter, Springer Verlag, February 2003 (*to appear*).
13. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In Reidel, D. Ordered Sets, Dordrecht, 1982, pp. 445–470.
14. Wille, R.: Concept lattices and conceptual knowledge. Computers and Mathematics with Applications **23** (1992) 493–522.

Author Index

- Baader, Franz 261
Bain, Michael 329
Becker, Peter 96, 172
Brawn, Peter 57
Busch, Peter 321
- Chen-Burger, Yun-Heh 252
Cigarrán, Juan M. 74
Cimiano, Philipp 189
Cole, Richard 172
- Dasmahapatra, Srinandan 252
Dau, Frithjof 156
Davey, Brian A. 55
Diatta, Jean 236
Ducrou, Jon 57
- Eklund, Peter 57
- Ferré, Sébastien 47
Freese, Ralph 112
Fu, Huaiguo 313, 394
Fu, Huaiyu 313
- Ganter, Bernhard 128
Godin, Robert 352
Gonzalo, Julio 74
Grigoriev, Peter A. 386
- Hereth Correia, Joachim 14, 39, 337
Hotho, Andreas 189
- Kaiser, Tim B. 39, 222
Kalfoglou, Yannis 252
King, Ross D. 47
Klinger, Julia 14
- Kourie, Derrick 372
Kuznetsov, Sergei O. 287
Kwuida, Léonard 142
- Martin, Ben 88
Mephu Nguifo, Engelbert 313, 394
Merwe, Dean van der 372
Missaoui, Rokia 352
- Njiwoua, Patrik 313
- Obiedkov, Sergei 372
Old, L. John 244
- Peñas, Anselmo 74
Pöschel, Reinhard 337
Priss, Uta 28
- Richards, Debbie 321, 402
- Schmidt, Stefan E. 222
Sertkaya, Baris 261
Stumme, Gerd 189
- Tane, Julien 189
Tilley, Thomas 104
- Valtchev, Petko 352
Verdejo, Felisa 74
Vormbrock, Björn 208
- Wille, Rudolf 1
Wolff, Karl Erich 180
- Yevtushenko, Serhiy A. 386