CLA 2015

Proceedings of the Twelfth International Conference on
Concept Lattices and Their Applications

CLA Conference Series

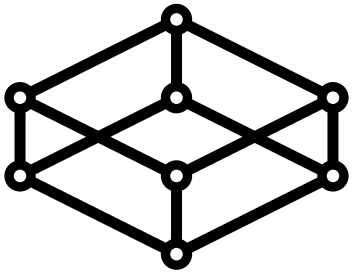
cla.inf.upol.cz

Blaise Pascal University, LIMOS laboratory, Clermont-Ferrand, France

ISBN 978–2–9544948–0–7

Blaise Pascal University, LIMOS laboratory, Clermont-Ferrand, France

The Twelfth International Conference on
Concept Lattices and Their Applications



CLA 2015

Clermont-Ferrand, France
October 13–16, 2015

Edited by

Sadok Ben Yahia
Jan Konecny

CLA 2015

Technical Editor:
Jan Konecny, jan.konecny@upol.cz

Cover photo from blognature.fr

Page count:	xiii+254
Impression:	50
Edition:	1st
First published:	2015

Published and printed by:
Blaise Pascal University, LIMOS laboratory, Clermont-Ferrand, France

Organization

CLA 2015 was organized by the Blaise Pascal University, LIMOS laboratory, Clermont-Ferrand.

Steering Committee

Radim Belohlavek	Palacký University, Olomouc, Czech Republic
Sadok Ben Yahia	Faculté des Sciences de Tunis, Tunisia
Jean Diatta	Université de la Réunion, France
Peter Eklund	IT University of Copenhagen, Denmark
Sergei O. Kuznetsov	State University HSE, Moscow, Russia
Engelbert Mephu Nguifo	LIMOS, University Blaise Pascal, Clermont-Ferrand, France
Amedeo Napoli	LORIA, Nancy, France
Manuel Ojeda-Aciego	Universidad de Málaga, Spain
Jan Outrata	Palacký University, Olomouc, Czech Republic

Program Chairs

Sadok Ben Yahia	Faculté des Sciences de Tunis, Tunis, Tunisia
Jan Konecny	Palacký University, Olomouc, Czech Republic

Program Committee

Simon Andrews	Sheffield Hallam University, Sheffield, United Kingdom
Jaume Baixeries	Universitat Politècnica de Catalunya, Barcelona, Catalonia
Radim Belohlavek	Palacký University, Olomouc, Czech Republic
Karell Bertet	L3i – Université de La Rochelle, La Rochelle, France
François Brucker	École Centrale, Marseille, France
Ana Burusco	Universidad Pública de Navarra, Pamplona, Spain
Claudio Carpineto	Fondazione Ugo Bordoni, Roma, Italy
Pablo Cordero	Universidad de Málaga, Málaga, Spain
Jean Diatta	Université de la Réunion, Saint-Denis, France
Felix Distel	Technische Universität Dresden, Dresden, Germany
Florent Domenach	University of Nicosia, Nicosia, Cyprus
Vincent Duquenne	Institut de Mathématiques de Jussieu, Paris, France
Peter Eklund	IT University of Copenhagen, Denmark
Sébastien Ferré	IRISA – Université de Rennes 1, Rennes, France
Bernhard Ganter	Technische Universität Dresden, Dresden, Germany

Cynthia Vera Glodeanu	Technische Universität Dresden, Dresden, Germany
Alain Gély	Université de Lorraine, Metz, France
Tarek Hamrouni	ISAMM, Manouba University, Tunisia
Marianne Huchard	LIRMM – Université Montpellier 2, Montpellier, France
Dmitry Ignatov	State University HSE, Moscow, Russia
Mehdi Kaytoue	Liris – Insa, Lyon, France
Stanislav Krajči	Univerzita Pavla Jozefa Šafárika v Košiciach, Košice, Slovakia
Francesco Kriegel	Technische Universität Dresden, Dresden, Germany
Michal Krupka	Palacký University, Olomouc, Czech Republic
Marzena Kryszkiewicz	Warsaw University of Technology, Warsaw, Poland
Sergei O. Kuznetsov	State University HSE, Moscow, Russia
Léonard Kwuida	Bern University of Applied Sciences, Bern, Switzerland
Jesús Medina	Universidad de Cádiz, Cádiz, Spain
Engelbert Mephu Nguifo	LIMOS, Clermont-Ferrand, France
Rokia Missaoui	LARIM – Université du Québec en Outaouais, Gatineau, Canada
Amedeo Napoli	LORIA, Nancy, France
Lhouari Nourine	LIMOS, Clermont-Ferrand, France
Sergei Obiedkov	State University HSE, Moscow, Russia
Manuel Ojeda-Aciego	Universidad de Málaga, Málaga, Spain
Petr Osička	Palacký University, Olomouc, Czech Republic
Jan Outrata	Palacký University, Olomouc, Czech Republic
Uta Priss	Ostfalia University of Applied Sciences, Wolfenbüttel, Germany
Francois Rioult	GREYC – Université de Caen Basse-Normandie, Caen, France
Sebastian Rudolph	Technische Universität Dresden, Dresden, Germany
Christian Sacarea	Babeș-Bolyai University, Cluj-Napoca, Romania
Barış Sertkaya	SAP Research Center, Dresden, Germany
László Szathmáry	University of Debrecen, Debrecen, Hungary
Petko Valtchev	Université du Québec, Montréal, Canada
Francisco Valverde	Universidad Carlos III, Madrid, Spain

Additional Reviewers

Ľubomír Antoni	Univerzita Pavla Jozefa Šafárika v Košiciach, Košice, Slovakia
Slim Bouker	LIMOS, Clermont-Ferrand, France
Maria Eugenia Cornejo Piñero	Universidad de Cádiz, Cádiz, Spain
Philippe Fournier-Viger	Université du Québec, Montréal, Canada
Eloisa Ramírez Poussa	Universidad de Cádiz, Cádiz, Spain
Stefan E. Schmidt	Technische Universität Dresden, Dresden, Germany
Vilém Vychodil	Palacký University, Olomouc, Czech Republic

Organization Committee

Olivier Raynaud (chair)	LIMOS, Clermont-Ferrand, France
Violaine Antoine	LIMOS, Clermont-Ferrand, France
Anne Berry	LIMOS, Clermont-Ferrand, France
Diyé Dia	LIMOS, Clermont-Ferrand, France
Kaoutar Ghazi	LIMOS, Clermont-Ferrand, France
Dhouha Grissa	INRA Theix, Clermont-Ferrand, France
Yannick Loiseau	LIMOS, Clermont-Ferrand, France
Engelbert Mephu Nguifo	LIMOS, Clermont-Ferrand, France
Séverine Miginiac	LIMOS, Clermont-Ferrand, France
Lhouari Nourine	LIMOS, Clermont-Ferrand, France

Table of Contents

Preface

Formal Concept Analysis is a method of analysis of logical data based on formalization of conceptual knowledge by means of lattice theory. It has proved to be of interest to various applied fields such as data visualization, knowledge discovery and data mining, database theory, and many others.

The International Conference "Concept Lattices and Their Applications (CLA)" is being organized since 2002 and its aim is to bring together researchers from various backgrounds to present and discuss their research related to FCA.

The Twelfth edition of CLA was held in Clermont-Ferrand, France from October 13 to 16, 2015. The event was jointly organized by the LIMOS laboratory, CNRS, and Blaise Pascal university, France.

This volume includes the selected papers and the abstracts of 5 invited talks. We would like to express our warmest thanks to the keynote speakers.

This year, there were initially 39 submissions, from which the Program Committee selected 20 papers which represents an acceptance rate of 51.2%.

The program of the conference consisted of five keynote talks given by the following distinguished researchers: Didier Dubois, Lhouari Nourine, Gabriella Pasi, Jan Ramon, and Takeaki Uno, together with twenty communications authored by researchers from 11 countries, namely: Austria, Czech republic, France, Germany, Kazakhstan, Republic of South Africa, Russia, Slovakia, Spain, Sweden, and Ukraine.

Each paper was reviewed by 3–4 members of the Program Committee and/or additional reviewers. We thank them all for their valuable assistance. It is planned that extended versions of the best papers will be published in a well-established journal, after another reviewing process.

The success of such event is mainly due the hard work and dedication of many people and a collaboration of several institutions. We thank the contributing authors, who submitted high quality works, we thank to the CLA Steering Committee, who gave us the opportunity of chairing this edition, and we thank the Program Committee, the additional reviewers, and the local Organization Committee. We are also thankful to the following institutions, which have helped the organization of the twelfth edition of CLA: Coffreo, IPLeanware, Axège, and Oorace.

We also thank the Easychair conference system as it made easier most of our administration tasks related to paper submission, selection, and reviewing. Last but not least we thank Jan Outrata, who offered his files for preparing the proceedings.

October 2015

Sadok Ben Yahia
Jan Konecny
Program Chairs of CLA 2015

User Models as Personal Ontologies

Gabriella Pasi

Laboratorio di Information Retrieval – Università degli Studi di Milano Bicocca, Milano, Italia

Abstract. The problem of defining user profiles has been a research issue since a long time; user profiles are employed in a variety of applications, including Information Filtering and Information Retrieval. In particular, considering the Information Retrieval task, user profiles are functional to the definition of approaches to Personalized search, which is aimed at tailoring the search outcome to users. In this context the quality of a user profile is clearly related to the effectiveness of the proposed personalized search solutions. A user profile represents the user interests and preferences; these can be captured either explicitly or implicitly. User profiles may be formally represented as bags of words, as vectors of words or concepts, or still as conceptual taxonomies. More recent approaches are aimed at formally representing user profiles as ontologies, thus allowing a richer, more structured and more expressive representation of the knowledge about the user.

This talk will address the issue of the automatic definition of personal ontologies, i.e. user-related ontologies. In particular, a method that applies a knowledge extraction process from the general purpose ontology YAGO will be described. Such a process is activated by a set of texts (or just a set of words) representatives of the user interests, and it is aimed to define a structured and semantically coherent representation of the user topical preferences. The issue of the evaluation of the generated representation will be discussed too.

Formal Concept Analysis from the Standpoint of Possibility Theory

Didier Dubois

IRIT – Université Paul Sabatier, Toulouse, France

Abstract. Formal concept analysis (FCA) and possibility theory (PoTh) are two theoretical frameworks that are addressing different concerns in the processing of information. Namely FCA builds concepts from a relation linking objects to the properties they satisfy, which has applications in data mining, clustering and related fields, while PoTh deals with the modeling of (graded) epistemic uncertainty. This difference of focus explains why the two settings have been developed completely independently for a very long time. However, it is possible to build a formal analogy between FCA and PoTh. Both theories heavily rely on the comparison of sets, in terms of containment or overlap. The four set-functions at work in PoTh actually determine all possible relative positions of two sets. Then the FCA operator defining the set of objects sharing a set of properties, which is at the basis of the definition of formal concepts, appears to be the counterpart of the set function expressing strong (or guaranteed) possibility in PoTh. Then, it suggests that the three other set functions existing in PoTh should also make sense in FCA, which leads to consider their FCA counterparts and new fixed point equations in terms of the new operators. One of these pairs of equations, paralleling the one defining formal concepts, define independent sub-contexts of objects and properties that have nothing in common.

The parallel of FCA with PoTh can still be made more striking using a cube of opposition (a device extending the traditional square of opposition existing in logic, and exhibiting a structure at work in many theories aiming at representing some aspects of the handling of information). The parallel of FCA with PoTh extends to conceptual pattern structures, where objects, may, e.g., be described by possibilistic knowledge bases.

In the talk we shall indicate various issues pertaining to FCA that could be worth studying in the future. For instance, the object-property links in formal contexts of FCA may be a matter of degree. These degrees may refer to very different notions, such as the degree of satisfaction of a gradual property, the degree of certainty that an object has, or not, a property, or still the typicality of an object with respect to a set of properties. These different intended semantics call for distinct manners of handling the degrees, as advocated in the presentation. Lastly, applications of FCA to the mining of association rules, to the fusion of conflicting pieces of information issued from multiple sources, to clustering of sets of objects on the basis of approximate concepts, or to the building of conceptual analogical proportions, will be discussed as other examples of lines of interest for further research.

Clarifying Lattice Structure by Data Polishing

Takeaki Uno

Institute of Informatics (NII) of Japan, Tokyo, Japan

Abstract. Concept lattice is made from many kinds of data. We want to use large scale data for the construction to capture wide and deep meanings but the result of the construction usually yields a quite huge lattice that is impossible to handle. Several techniques have been proposed to cope with this problem, but to best of our knowledge no algorithm attains good granularity, coverage, size distribution, and independence of concepts at the same time. We consider this difficulty comes from that the concepts are not clear in the data, so a good approach is to clarify the concepts in the data by some operations. In this direction, we propose "data polishing" that modify the data according to feasible hypothesis so that the concepts becomes clear.

Tractable Interesting Pattern Mining in Large Networks

Jan Ramon

University of Leuven – INRIA, Leuven, Belgium

Abstract. Pattern mining is an important data mining task. While the simplest setting, itemset mining, has been thoroughly studied real-world data is getting increasingly complex and network structured, e.g. in the context of social networks, economic networks, traffic networks, administrative networks, chemical interaction networks and biological regulatory networks.

This presentation will first provide an overview of graph pattern mining work, and will then discuss two important questions. First, what is an interesting concept, and can we obtain suitable mathematical properties to order concepts in some way, obtaining a lattice or other exploitable structure?

Second, how can we extract collections of interesting patterns from network-structured data in a computationally tractable way? In the case of graphs, having a lattice on the class of patterns turns out to be insufficient for computational tractability. We will discuss difficulties related to pattern matching and related to enumeration, and additional difficulties arising when considering condensed pattern mining variants.

Extended Dualization: Application to Maximal Pattern Mining

Lhouari Nourine

Limos, Clermont-Ferrand, France

Abstract. The hypergraph dualization is a crucial step in many applications in logics, databases, artficial intelligence and pattern mining, especially for hypergraphs or boolean lattices. The objective of this talk is to study polynomial reductions of the dualization problem on arbitrary posets to the dualization problem on boolean lattices, for which output quasi-polynomial time algorithms exist.

The main application domain concerns pattern mining problems, i.e. the identification of maximal interesting patterns in database by asking membership queries (predicate) to a database.

Subset-generated complete sublattices as concept lattices*

Martin Kauer and Michal Krupka

Department of Computer Science
Palacký University in Olomouc
Czech Republic
martin.kauer@upol.cz
michal.krupka@upol.cz

Abstract. We present a solution to the problem of finding the complete sublattice of a given concept lattice generated by given set of elements. We construct the closed subrelation of the incidence relation of the corresponding formal context whose concept lattice is equal to the desired complete sublattice. The construction does not require the presence of the original concept lattice. We introduce an efficient algorithm for the construction and give an example and experiments.

1 Introduction and problem statement

One of the basic theoretical results of Formal Concept Analysis (FCA) is the correspondence between closed subrelations of a formal context and complete sublattices of the corresponding concept lattice [2]. In this paper, we study a related problem of constructing the closed subrelation for a complete sublattice generated by given set of elements.

Let $\langle X, Y, I\rangle$ be a formal context, $\mathcal{B}(X, Y, I)$ its concept lattice. Denote by V the complete sublattice of $\mathcal{B}(X, Y, I)$ generated by a set $P \subseteq \mathcal{B}(X, Y, I)$. As it is known [2], there exists a closed subrelation $J \subseteq I$ with the concept lattice $\mathcal{B}(X, Y, J)$ equal to V. We show a method of constructing J without the need of constructing $\mathcal{B}(X, Y, I)$ first. We also provide an efficient algorithm (with polynomial time complexity), implementing the method. The paper also contains an illustrative example and results of experiments, performed on the *Mushroom* dataset from the UCI Machine Learning Repository.

2 Complete lattices and Formal Concept Analysis

Recall that a partially ordered set U is called a *complete lattice* if each its subset $P \subseteq U$ has a supremum and infimum. We denote these by $\bigvee P$ and $\bigwedge P$, respectively. A subset $V \subseteq U$ is a $\bigvee$*-subsemilattice* (resp. $\bigwedge$*-subsemilattice*, resp. *complete sublattice*) *of* U, if for each $P \subseteq V$ it holds $\bigvee P \in V$ (resp. $\bigwedge P \in V$, resp. $\{\bigvee P, \bigwedge P\} \subseteq V$).

* Supported by the IGA of Palacký University Olomouc, No. PrF 2015 023

For a subset $P \subseteq U$ we denote by $\mathrm{C}_{\bigvee} P$ the $\bigvee$-subsemilattice of U *generated by* P, i.e. the smallest (w.r.t. set inclusion) $\bigvee$-subsemilattice of U containing P. $\mathrm{C}_{\bigvee} P$ always exists and is equal to the intersection of all $\bigvee$-subsemilattices of U containing P. The $\bigwedge$-subsemilattice of U generated by P and the complete sublattice of U generated by P are defined similarly and are denoted by $\mathrm{C}_{\bigwedge} P$ and $\mathrm{C}_{\bigvee\bigwedge} P$, respectively.

The operators $\mathrm{C}_{\bigvee}$, $\mathrm{C}_{\bigwedge}$, and $\mathrm{C}_{\bigvee\bigwedge}$ are closure operators on the set U. Recall that a *closure operator on a set* X is a mapping $\mathrm{C}\colon 2^X \to 2^X$ (where 2^X is the set of all subsets of X) satisfying for all sets $A, A_1, A_2 \subseteq X$

1. $A \subseteq \mathrm{C}\, A$,
2. if $A_1 \subseteq A_2$ then $\mathrm{C}\, A_1 \subseteq \mathrm{C}\, A_2$,
3. $\mathrm{CC}\, A = \mathrm{C}\, A$.

Concept lattices have been introduced in [4], our basic reference is [2]. A (*formal*) *context* is a triple $\langle X, Y, I\rangle$ where X is a set of objects, Y a set of attributes and $I \subseteq X \times Y$ a binary relation between X and Y specifying for each object which attributes it has.

For subsets $A \subseteq X$ and $B \subseteq Y$ we set

$$A^{\uparrow_I} = \{y \in Y \mid \text{for each } x \in A \text{ it holds } \langle x, y\rangle \in I\},$$
$$B^{\downarrow_I} = \{x \in X \mid \text{for each } y \in B \text{ it holds } \langle x, y\rangle \in I\}.$$

The pair $\langle {}^{\uparrow_I}, {}^{\downarrow_I}\rangle$ is a Galois connection between sets X and Y, i.e. it satisfies

1. If $A_1 \subseteq A_2$ then $A_2^{\uparrow_I} \subseteq A_1^{\uparrow_I}$, if $B_1 \subseteq B_2$ then $B_2^{\downarrow_I} \subseteq B_1^{\downarrow_I}$.
2. $A \subseteq A^{\uparrow_I\downarrow_I}$ and $B \subseteq B^{\downarrow_I\uparrow_I}$.

The operator ${}^{\uparrow_I\downarrow_I}$ is a closure operator on X and the operator ${}^{\downarrow_I\uparrow_I}$ is a closure operator on Y.

A pair $\langle A, B\rangle$ satisfying $A^{\uparrow_I} = B$ and $B^{\downarrow_I} = A$ is called a (*formal*) *concept* of $\langle X, Y, I\rangle$. The set A is then called *the extent of* $\langle A, B\rangle$, the set B *the intent of* $\langle A, B\rangle$. When there is no danger of confusion, we can use the term "an extent of I" instead of "the extent of a concept of $\langle X, Y, I\rangle$", and similarly for intents.

A partial order $\leq$ on the set $\mathcal{B}(X, Y, I)$ of all formal concepts of $\langle X, Y, I\rangle$ is defined by $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$ iff $A_1 \subseteq A_2$ (iff $B_2 \subseteq B_1$). $\mathcal{B}(X, Y, I)$ along with $\leq$ is a complete lattice and is called *the concept lattice of* $\langle X, Y, I\rangle$. Infima and suprema in $\mathcal{B}(X, Y, I)$ are given by

$$\bigwedge_{j\in J} \langle A_j, B_j\rangle = \left\langle \bigcap_{j\in J} A_j, \left(\bigcup_{j\in J} B_j\right)^{\downarrow_I\uparrow_I} \right\rangle, \tag{1}$$

$$\bigvee_{j\in J} \langle A_j, B_j\rangle = \left\langle \left(\bigcup_{j\in J} A_j\right)^{\uparrow_I\downarrow_I}, \bigcap_{j\in J} B_j \right\rangle. \tag{2}$$

One of immediate consequences of (1) and (2) is that the intersection of any system of extents, resp. intents, is again an extent, resp. intent, and that it can be expressed as follows:

$$\bigcap_{j\in J} B_j = \left(\bigcup_{j\in J} A_j\right)^{\uparrow_I}, \quad \text{resp.} \quad \bigcap_{j\in J} A_j = \left(\bigcup_{j\in J} B_j\right)^{\downarrow_I},$$

for concepts $\langle A_j, B_j\rangle \in \mathcal{B}(X,Y,I)$, $j \in J$.

Concepts $\langle \{y\}^{\downarrow_I}, \{y\}^{\downarrow_I\uparrow_I}\rangle$ where $y \in Y$ are *attribute concepts*. Each concept $\langle A, B\rangle$ is infimum of some attribute concepts (we say the set of all attribute concepts is $\bigwedge$*-dense in* $\mathcal{B}(X,Y,I)$). More specifically, $\langle A, B\rangle$, is infimum of attribute concepts $\langle \{y\}^{\downarrow_I}, \{y\}^{\downarrow_I\uparrow_I}\rangle$ for $y \in B$ and $A = \bigcap_{y\in B}\{y\}^{\downarrow_I}$.

Dually, concepts $\langle \{x\}^{\uparrow_I\downarrow_I}, \{x\}^{\uparrow_I}\rangle$ for $x \in X$ are *object concepts*, they are $\bigvee$-dense in $\mathcal{B}(X,Y,I)$ and for each concept $\langle A, B\rangle$, $B = \bigcap_{x\in A}\{x\}^{\uparrow_I}$.

A subrelation $J \subseteq I$ is called a *closed subrelation of* I if each concept of $\langle X, Y, J\rangle$ is also a concept of $\langle X, Y, I\rangle$. There is a correspondence between closed subrelations of I and complete sublattices of $\mathcal{B}(X,Y,I)$ [2, Theorem 13]: For each closed subrelation $J \subseteq I$, $\mathcal{B}(X,Y,J)$ is a complete sublattice of $\mathcal{B}(X,Y,I)$, and to each complete sublattice $V \subseteq \mathcal{B}(X,Y,I)$ there exists a closed subrelation $J \subseteq I$ such that $V = \mathcal{B}(X,Y,J)$.

3 Closed subrelations for generated sublattices

Let us have a context $\langle X, Y, I\rangle$ and a subset P of its concept lattice. Denote by V the complete sublattice of $\mathcal{B}(X,Y,I)$ generated by P (i.e. $V = \mathrm{C}_{\bigvee\bigwedge}P$). Our aim is to find, without computing the lattice $\mathcal{B}(X,Y,I)$, the closed subrelation $J \subseteq I$ whose concept lattice $\mathcal{B}(X,Y,J)$ is equal to V.

If $\mathcal{B}(X,Y,I)$ is finite, V can be obtained by alternating applications of the closure operators $\mathrm{C}_{\bigvee}$ and $\mathrm{C}_{\bigwedge}$ to P: we set $V_1 = \mathrm{C}_{\bigvee}P$, $V_2 = \mathrm{C}_{\bigwedge}V_1$, ..., and, generally, $V_i = \mathrm{C}_{\bigvee}V_{i-1}$ for odd $i > 1$ and $V_i = \mathrm{C}_{\bigwedge}V_{i-1}$ for even $i > 1$. The sets V_i are $\bigvee$-subsemilattices (for odd i) resp. $\bigwedge$-subsemilattices (for even i) of $\mathcal{B}(X,Y,I)$. Once $V_i = V_{i-1}$, we have the complete sublattice V.

Note that for infinite $\mathcal{B}(X,Y,I)$, V can be infinite even if P is finite. Indeed, denoting $FL(P)$ the free lattice generated by P [3] and setting $X = Y = FL(P)$, $I = \leq$ we have $FL(P) \subseteq V \subseteq \mathcal{B}(X,Y,I)$. ($\mathcal{B}(X,Y,I)$ is the Dedekind-MacNeille completion of $FL(P)$ [2], and we identify P and $FL(P)$ with subsets of $\mathcal{B}(X,Y,I)$ as usual.) Now, if $|P| > 2$ then $FL(P)$ is infinite [3], and so is V.

We always consider sets V_i together with the appropriate restriction of the ordering on $\mathcal{B}(X,Y,I)$. For each $i > 0$, V_i is a complete lattice (but not a complete sublattice of $\mathcal{B}(X,Y,I)$).

In what follows, we construct formal contexts with concept lattices isomorphic to the complete lattices V_i, $i > 0$. First, we find a formal context for the complete lattice V_1. Let $K_1 \subseteq P \times Y$ be given by

$$\langle\langle A, B\rangle, y\rangle \in K_1 \quad \text{iff} \quad y \in B. \tag{3}$$

As we can see, rows in the context $\langle P, Y, K_1 \rangle$ are exactly intents of concepts from P.

Proposition 1. *The concept lattice $\mathcal{B}(P, Y, K_1)$ and the complete lattice V_1 are isomorphic. The isomorphism assigns to each concept $\langle B^{\downarrow K_1}, B \rangle \in \mathcal{B}(P, Y, K_1)$ the concept $\langle B^{\downarrow I}, B \rangle \in \mathcal{B}(X, Y, I)$.*

Proof. Concepts from V_1 are exactly those with intents equal to intersections of intents of concepts from P. The same holds for concepts from $\mathcal{B}(P, Y, K_1)$. □

Next we describe formal contexts for complete lattices V_i, $i > 1$. All of the contexts are of the form $\langle X, Y, K_i \rangle$, i.e. they have the set X as the set of objects and the set Y as the set of attributes (the relation K_1 is different in this regard). The relations K_i for $i > 1$ are defined in a recursive manner:

$$\text{for } i > 1,\ \langle x, y \rangle \in K_i \quad \text{iff} \quad \begin{cases} x \in \{y\}^{\downarrow K_{i-1} \uparrow K_{i-1} \downarrow I} \text{ for even } i, \\ y \in \{x\}^{\uparrow K_{i-1} \downarrow K_{i-1} \uparrow I} \text{ for odd } i. \end{cases} \tag{4}$$

Proposition 2. *For each $i > 1$,*

1. $K_i \subseteq I$,
2. $K_i \subseteq K_{i+1}$.

Proof. We will prove both parts for odd i; the assertions for even i are proved similarly.

1. Let $\langle x, y \rangle \in K_i$. From $\{y\} \subseteq \{y\}^{\downarrow K_{i-1} \uparrow K_{i-1}}$ we get $\{y\}^{\downarrow K_{i-1} \uparrow K_{i-1} \downarrow I} \subseteq \{y\}^{\downarrow I}$. Thus, $x \in \{y\}^{\downarrow K_{i-1} \uparrow K_{i-1} \downarrow I}$ implies $x \in \{y\}^{\downarrow I}$, which is equivalent to $\langle x, y \rangle \in I$.

2. As $K_i \subseteq I$, we have $\{y\}^{\downarrow K_i \uparrow K_i \downarrow I} \supseteq \{y\}^{\downarrow K_i \uparrow K_i \downarrow K_i} = \{y\}^{\downarrow K_i}$. Thus, $x \in \{y\}^{\downarrow K_i}$ yields $x \in \{y\}^{\downarrow K_i \uparrow K_i \downarrow I}$. □

We can see that the definitions of K_i for even and odd $i > 1$ are dual. In what follows, we prove properties of K_i for even i and give the versions for odd i without proofs.

First we give two basic properties of K_i that are equivalent to the definition. The first one says that K_i can be constructed as a union of some specific rectangles, the second one will be used frequently in what follows.

Proposition 3. *Let $i > 1$.*

1. *If i is even then $K_i = \bigcup_{y \in Y} \{y\}^{\downarrow K_{i-1} \uparrow K_{i-1} \downarrow I} \times \{y\}^{\downarrow K_{i-1} \uparrow K_{i-1}}$. If i is odd then $K_i = \bigcup_{x \in X} \{x\}^{\uparrow K_{i-1} \downarrow K_{i-1} \uparrow I} \times \{x\}^{\uparrow K_{i-1} \downarrow K_{i-1}}$.*
2. *If i is even then for each $y \in Y$, $\{y\}^{\downarrow K_i} = \{y\}^{\downarrow K_{i-1} \uparrow K_{i-1} \downarrow I}$. If i is odd then for each $x \in X$, $\{x\}^{\uparrow K_i} = \{x\}^{\uparrow K_{i-1} \downarrow K_{i-1} \uparrow I}$.*

Proof. We will prove only the assertions for even i.

1. The "$\subseteq$" inclusion is evident. We will prove the converse inclusion. If $\langle x, y \rangle \in \bigcup_{y' \in Y} \{y'\}^{\downarrow K_{i-1} \uparrow K_{i-1} \downarrow I} \times \{y'\}^{\downarrow K_{i-1} \uparrow K_{i-1}}$ then there is $y' \in Y$ such that $x \in \{y'\}^{\downarrow K_{i-1} \uparrow K_{i-1} \downarrow I}$ and $y \in \{y'\}^{\downarrow K_{i-1} \uparrow K_{i-1}}$. The latter implies $\{y\}^{\downarrow K_{i-1} \uparrow K_{i-1}} \subseteq$

$\{y'\}^{\downarrow K_{i-1}\uparrow K_{i-1}}$, whence $\{y'\}^{\downarrow K_{i-1}\uparrow K_{i-1}\downarrow I} \subseteq \{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}\downarrow I}$. Thus, x belongs to $\{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}\downarrow I}$ and by definition, $\langle x, y\rangle \in K_i$.

2. Follows directly from the obvious fact that $x \in \{y\}^{\downarrow K_i}$ if and only if $\langle x, y\rangle \in K_i$. □

A direct consequence of 2. of Prop. 3 is the following.

Proposition 4. *If i is even then each extent of K_i is also an extent of I. If i is odd then each intent of K_i is also an intent of I.*

Proof. Let i be even. 2. of Prop. 3 implies that each attribute extent of K_i is an extent of I. Thus, the proposition follows from the fact that each extent of K_i is an intersection of attribute extents of K_i.

The statement for odd i is proved similarly except for $i = 1$ where it follows by definition. □

Proposition 5. *Let $i > 1$. If i is even then for each $y \in Y$ it holds*

$$\{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}} = \{y\}^{\downarrow K_i\uparrow K_i} = \{y\}^{\downarrow K_i\uparrow I}.$$

If i is odd then for each $x \in X$ we have

$$\{x\}^{\uparrow K_{i-1}\downarrow K_{i-1}} = \{x\}^{\uparrow K_i\downarrow K_i} = \{x\}^{\uparrow K_i\downarrow I}.$$

Proof. We will prove the assertion for even i. By Prop. 4, $\{y\}^{\downarrow K_i}$ is an extent of I. The corresponding intent is

$$\{y\}^{\downarrow K_i\uparrow I} = \{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}\downarrow I\uparrow I} = \{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}} \tag{5}$$

(by Prop. 4, $\{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}}$ is an intent of I). Moreover, as $K_i \subseteq I$ (Prop. 2), we have

$$\{y\}^{\downarrow K_i\uparrow K_i} \subseteq \{y\}^{\downarrow K_i\uparrow I}. \tag{6}$$

We prove $\{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}} \subseteq \{y\}^{\downarrow K_i\uparrow K_i}$. Let $y' \in \{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}}$. It holds

$$\{y'\}^{\downarrow K_{i-1}\uparrow K_{i-1}} \subseteq \{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}}$$

($^{\downarrow K_{i-1}\uparrow K_{i-1}}$ is a closure operator). Thus, $\{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}\downarrow I} \subseteq \{y'\}^{\downarrow K_{i-1}\uparrow K_{i-1}\downarrow I}$ and so by 2. of Prop. 3, $\{y\}^{\downarrow K_i} \subseteq \{y'\}^{\downarrow K_i}$. Applying $^{\uparrow K_i}$ to both sides we obtain $\{y'\}^{\downarrow K_i\uparrow K_i} \subseteq \{y\}^{\downarrow K_i\uparrow K_i}$ proving $y' \in \{y\}^{\downarrow K_i\uparrow K_i}$.

This, together with (5) and (6), proves the proposition. □

Proposition 6. *Let $i > 1$ be even. Then for each intent B of K_{i-1} it holds $B^{\downarrow K_i} = B^{\downarrow I}$. Moreover, if B is an attribute intent (i.e. there is $y \in Y$ such that $B = \{y\}^{\downarrow K_{i-1}\uparrow K_{i-1}}$) then $\langle B^{\downarrow K_i}, B\rangle$ is a concept of I.*

If $i > 1$ is odd then for each extent A of K_{i-1} it holds $A^{\uparrow K_i} = A^{\uparrow I}$. If A is an object extent (i.e. there is $x \in X$ such that $A = \{x\}^{\uparrow K_{i-1}\downarrow K_{i-1}}$) then $\langle A, A^{\uparrow K_i}\rangle$ is a concept of I.

Proof. We will prove the assertion for even i. Let B be an intent of K_{i-1}. It holds $B = \bigcup_{y\in B}\{y\}$ (obviously) and hence $B = \bigcup_{y\in B}\{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}}$ (since $^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}}$ is a closure operator). Therefore (2. of Prop. 3),

$$B^{\downarrow_{K_i}} = \left(\bigcup_{y\in B}\{y\}\right)^{\downarrow_{K_i}} = \bigcap_{y\in B}\{y\}^{\downarrow_{K_i}} = \bigcap_{y\in B}\{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}\downarrow_I}$$
$$= \left(\bigcup_{y\in B}\{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}}\right)^{\downarrow_I} = B^{\downarrow_I},$$

proving the first part.

Now let B be an attribute intent of K_{i-1}, $B = \{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}}$. By 2. of Prop. 3 it holds $B^{\downarrow_I} = \{y\}^{\downarrow_{K_i}}$. By Prop. 5, $B^{\downarrow_I\uparrow_I} = \{y\}^{\downarrow_{K_i}\uparrow_I} = \{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}} = B$. □

Now we turn to complete lattices V_i defined above. We have already shown in Prop. 1 that the complete lattice V_1 and the concept lattice $\mathcal{B}(P, Y, K_1)$ are isomorphic. Now we give a general result for $i > 0$.

Proposition 7. *For each $i > 0$, the concept lattice $\mathcal{B}(P, Y, K_i)$ (for $i = 1$) resp. $\mathcal{B}(X, Y, K_i)$ (for $i > 1$) and the complete lattice V_i are isomorphic. The isomorphism is given by $\langle B^{\downarrow_{K_i}}, B\rangle \mapsto \langle B^{\downarrow_I}, B\rangle$ if i is odd and by $\langle A, A^{\uparrow_{K_i}}\rangle \mapsto \langle A, A^{\uparrow_I}\rangle$ if i is even.*

Proof. We will proceed by induction on i. The base step $i = 1$ has been already proved in Prop. 1. We will do the induction step for even i, the other case is dual.

As $V_i = \mathrm{C}_{\bigwedge} V_{i-1}$, we have to

1. show that the set $W = \{\langle A, A^{\uparrow_I}\rangle \mid A \text{ is an extent of } K_i\}$ is a subset of $\mathcal{B}(X, Y, I)$, containing V_{i-1} and
2. find for each $\langle A, A^{\uparrow_{K_i}}\rangle \in \mathcal{B}(X, Y, K_i)$ a set of concepts from V_{i-1} whose infimum in $\mathcal{B}(X, Y, I)$ has extent equal to A.

1. By Prop. 4, each extent of K_i is also an extent of I. Thus, $W \subseteq \mathcal{B}(X, Y, I)$. If $\langle A, B\rangle \in V_{i-1}$ then by the induction hypothesis B is an intent of K_{i-1} ($i - 1$ is odd). By Prop. 6, $B^{\downarrow_{K_i}} = B^{\downarrow_I} = A$ is an extent of K_i and so $\langle A, B\rangle \in W$.

2. Denote $B = A^{\uparrow_{K_i}}$. For each $y \in Y$, $\{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}}$ is an intent of K_{i-1}. By Prop. 3 and the induction hypothesis,

$$\langle \{y\}^{\downarrow_{K_i}}, \{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}}\rangle = \langle \{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}\downarrow_I}, \{y\}^{\downarrow_{K_{i-1}}\uparrow_{K_{i-1}}}\rangle \in V_{i-1}.$$

Now, the extent of the infimum (taken in $\mathcal{B}(X, Y, I)$) of these concepts for $y \in B$ is equal to $\bigcap_{y\in B}\{y\}^{\downarrow_{K_i}} = B^{\downarrow_{K_i}} = A$. □

If X and Y are finite then 2. of Prop. 2 implies there is a number $n > 1$ such that $K_{n+1} = K_n$. Denote this relation by J. According to Prop. 7, there are two isomorphisms of the concept lattice $\mathcal{B}(X, Y, J)$ and $V_n = V_{n+1} = V$. We will show that these two isomorphisms coincide and $\mathcal{B}(X, Y, J)$ is actually equal to V. This will also imply J is a closed subrelation of I.

Proposition 8. $\mathcal{B}(X, Y, J) = V$.

Proof. Let $\langle A, B\rangle \in \mathcal{B}(X, Y, J)$. It suffices to show that $\langle A, B\rangle \in \mathcal{B}(X, Y, I)$. As $J = K_{n+1} = K_n$ we have $J = K_i$ for some even i and also $J = K_i$ for some odd i. We can therefore apply both parts of Prop. 6 to J obtaining $A = B^{\downarrow_J} = B^{\downarrow_I}$ and $B = A^{\uparrow_J} = A^{\uparrow_I}$. □

Algorithm 1 uses our results to compute the subrelation J for given $\langle X, Y, I\rangle$ and P.

Algorithm 1 Computing the closed subrelation J.

Input: formal context $\langle X, Y, I\rangle$, subset $P \subseteq \mathcal{B}(X, Y, I)$
Output: the closed subrelation of $J \subseteq I$ whose concept lattice is equal to $\mathrm{C}_{\bigvee\bigwedge} P$

```
J ← relation K_1 (3)
i ← 1
repeat
    L ← J
    i ← i + 1
    if i is even then
        J ← {⟨x, y⟩ ∈ X × Y | x ∈ {y}^{↓_L ↑_L ↓_I}}
    else
        J ← {⟨x, y⟩ ∈ X × Y | y ∈ {x}^{↑_L ↓_L ↑_I}}
    end if
until i > 2 & J = L
return J
```

Proposition 9. *Algorithm 1 is correct and terminates after at most* $\max(|I| + 1, 2)$ *iterations.*

Proof. Correctness follows from Prop. 8. The terminating condition ensures we compare J and L only when they are both subrelations of the context $\langle X, Y, I\rangle$ (after the first iteration, L is a subrelation of $\langle P, Y, K_1\rangle$ and the comparison would not make sense).

After each iteration, L holds the relation K_{i-1} and J holds K_i (4). Thus, except for the first iteration, we have $L \subseteq J$ before the algorithm enters the terminating condition (Prop. 2). As J is always a subset of I (Prop. 2), the number of iterations will not be greater than $|I| + 1$. The only exception is $I = \emptyset$. In this case, the algorithm will terminate after 2 steps due to the first part of the terminating condition. □

4 Examples and experiments

Let $\langle X, Y, I\rangle$ be the formal context from Fig. 1 (left). The associated concept lattice $\mathcal{B}(X, Y, I)$ is depicted in Fig. 1 (right). Let $P = \{c_1, c_2, c_3\}$ where

I	y_1	y_2	y_3	y_4	y_5
x_1	×			×	
x_2	×	×	×		
x_3			×		×
x_4			×		
x_5		×			

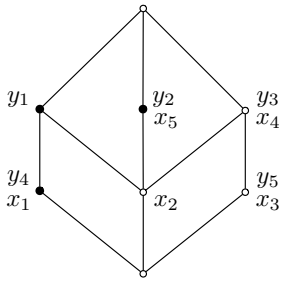
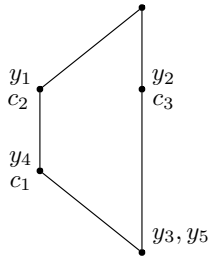


Fig. 1: Formal context $\langle X, Y, I\rangle$ (left) and concept lattice $\mathcal{B}(X, Y, I)$, together with a subset $P \subseteq \mathcal{B}(X, Y, I)$, depicted by filled dots (right).

$c_1 = \langle\{x_1\}, \{y_1, y_4\}\rangle$, $c_2 = \langle\{x_1, x_2\}, \{y_1\}\rangle$, $c_3 = \langle\{x_2, x_5\}, \{y_2\}\rangle$ are concepts from $\mathcal{B}(X, Y, I)$. These concept are depicted in Fig. 1 by filled dots.

First, we construct the context $\langle P, Y, K_1\rangle$ (3). Rows in this context are intents of concepts from P (see Fig. 2, left). The concept lattice $\mathcal{B}(P, Y, K_1)$ (Fig. 2, center) is isomorphic to the $\bigvee$-subsemilattice $V_1 = \mathrm{C}_{\bigvee} P \subseteq \mathcal{B}(X, Y, I)$ (Fig. 2, right). It is easy to see that elements of $\mathcal{B}(P, Y, K_1)$ and corresponding elements

K_1	y_1	y_2	y_3	y_4	y_5
c_1	×			×	
c_2	×				
c_3		×			



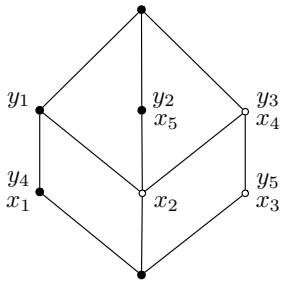


Fig. 2: Formal context $\langle P, Y, K_1\rangle$ (left), the concept lattice $\mathcal{B}(P, Y, K_1)$ (center) and the $\bigvee$-subsemilattice $\mathrm{C}_{\bigvee} P \subseteq \mathcal{B}(X, Y, I)$, isomorphic to $\mathcal{B}(P, Y, K_1)$, depicted by filled dots (right).

of V_1 have the same intents.

Next step is to construct the subrelation $K_2 \subseteq I$. By (4), K_2 consists of elements $\langle x, y\rangle \in X \times Y$ satisfying $x \in \{y\}^{\downarrow_{K_1}\uparrow_{K_1}\downarrow_I}$. The concept lattice $\mathcal{B}(X, Y, K_2)$ is isomorphic to the $\bigwedge$-subsemilattice $V_2 = \mathrm{C}_{\bigwedge} V_1 \subseteq \mathcal{B}(X, Y, I)$. K_2, $\mathcal{B}(X, Y, K_2)$, and V_2 are depicted in Fig. 3.

The subrelation $K_3 \subseteq I$ is computed again by (4). K_3 consists of elements $\langle x, y\rangle \in X \times Y$ satisfying $y \in \{x\}^{\uparrow_{K_2}\downarrow_{K_2}\uparrow_I}$. The result can be viewed in Fig. 4.

K_2	y_1	y_2	y_3	y_4	y_5
x_1	×			×	
x_2	×	×	·		
x_3			·		·
x_4			·		
x_5		×			

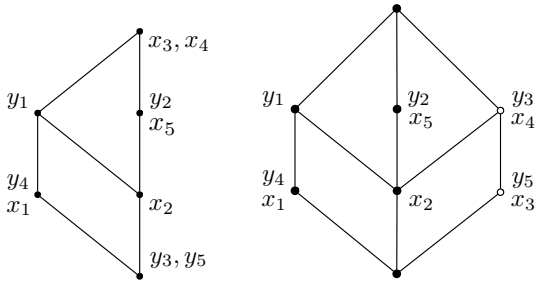


Fig. 3: Formal context $\langle X, Y, K_2 \rangle$ (left), the concept lattice $\mathcal{B}(X, Y, K_2)$ (center) and the $\bigwedge$-subsemilattice $V_2 = \mathrm{C}_{\bigwedge} V_1 \subseteq \mathcal{B}(X, Y, I)$, isomorphic to $\mathcal{B}(X, Y, K_2)$, depicted by filled dots (right). Elements of $I \setminus K_2$ are depicted by dots in the table.

K_3	y_1	y_2	y_3	y_4	y_5
x_1	×			×	
x_2	×	×	×		
x_3			·		·
x_4			·		
x_5		×			

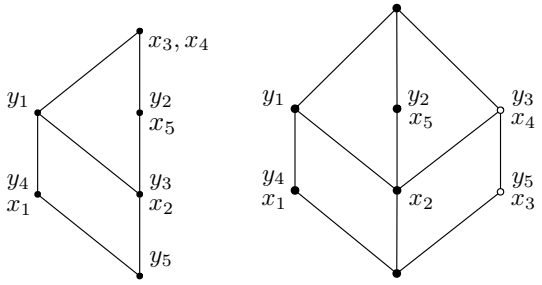


Fig. 4: Formal context $\langle X, Y, K_3 \rangle$ (left), the concept lattice $\mathcal{B}(X, Y, K_3)$ (center) and the $\bigvee$-subsemilattice $V_3 = \mathrm{C}_{\bigvee} V_2 \subseteq \mathcal{B}(X, Y, I)$, isomorphic to $\mathcal{B}(X, Y, K_3)$, depicted by filled dots (right). Elements of $I \setminus K_3$ are depicted by dots in the table. As $K_3 = K_4 = J$, it is a closed subrelation of I and $V_4 = \mathrm{C}_{\bigwedge} V_3 = V_3$ is a complete sublattice of $\mathcal{B}(X, Y, I)$.

Notice that already $V_3 = V_2$ but $K_3 \neq K_2$. We cannot stop and have to perform another step. After computing K_4 we can easily check that $K_4 = K_3$. We thus obtained the desired closed subrelation $J \subseteq I$ and $V_4 = V_3$ is equal to the desired complete sublattice $V \subseteq \mathcal{B}(X, Y, I)$.

In [1], the authors present an algorithm for computing a sublattice of a given lattice generated by a given set of elements. Originally, we planned to include a comparison between their approach and our Alg. 1. Unfortunately, the algorithm in [1] turned out to be incorrect. It is based on the false claim that (using our notation) the smallest element of V, which is greater than or equal to an element $v \in \mathcal{B}(X, Y, I)$, is equal to $\bigwedge\{p \in P \mid p \geq v\}$. The algorithm from [1] fails e.g. on the input depicted in Fig. 5.

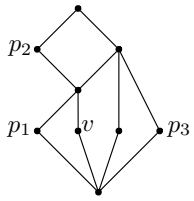


Fig. 5: An example showing that the algorithm from [1] is incorrect. A complete lattice with a selected subset $P = \{p_1, p_2, p_3\}$. The least element of the sublattice V generated by P which is greater than or equal to v is $p_1 \vee v$. The algorithm incorrectly chooses p_2 and "forgets" to add $p_1 \vee v$ to the output.

The time complexity of our algorithm is clearly polynomial w.r.t. $|X|$ and $|Y|$. In Prop. 9 we proved that the number of iterations is $\mathcal{O}(|I|)$. Our experiments indicate that this number might be much smaller in the practice. We used the *Mushroom* dataset from the UC Irvine Machine Learning Repository, which contains 8124 objects, 119 attributes and 238710 concepts. For 39 different sizes of the set P, we selected randomly its elements, 1000 times for each of the sizes. For each P, we ran our algorithm and measured the number n of iterations, after which the algorithm terminated. We can see in Tbl. 1 maximal and average values of n, separately for each size of P. From the results in Tbl. 1 we can see

$\|P\|$(%)	Max n	Avg n	$\|P\|$(%)	Max n	Avg n	$\|P\|$(%)	Max n	Avg n
0.005	11	7	0.25	6	3	0.90	5	3
0.010	10	6	0.30	6	3	0.95	4	3
0.015	10	5	0.35	6	3	1	4	3
0.020	10	5	0.40	5	3	2	4	3
0.025	8	5	0.45	5	3	3	4	3
0.030	8	4	0.50	5	3	4	4	3
0.035	8	4	0.55	6	3	5	4	2
0.040	7	4	0.60	5	3	6	4	2
0.045	10	4	0.65	4	3	7	4	2
0.050	8	4	0.70	5	3	8	3	2
0.100	6	4	0.75	6	3	9	3	2
0.150	6	4	0.80	6	3	10	3	2
0.200	6	4	0.85	4	3	11	3	2

Table 1: Results of experiments on *Mushrooms* dataset. The size of P is given by the percentage of the size of the concept lattice.

that the number of iterations (both maximal and average values) is very small compared to the number of objects and attributes. There is also an apparent decreasing trend of number of iterations for increasing size of P.

5 Conclusion and open problems

An obvious advantage of our approach is that we avoid computing the whole concept lattice $\mathcal{B}(X, Y, I)$. This should lead to shorter computation time, especially if the generated sublattice V is substantially smaller than $\mathcal{B}(X, Y, I)$.

The following is an interesting observation and an open problem. It is mentioned in [2] that the system of all closed subrelations of I is not a closure system and, consequently, there does not exist a closure operator assigning to each subrelation of I a least greater (w.r.t. set inclusion) closed subrelation. This is indeed true as the intersection of closed subrelations need not be a closed subrelation. However, our method can be easily modified to compute for any subrelation $K \subseteq I$ a closed subrelation $J \supseteq K$, which seems to be minimal in some sense. Indeed, we can set $K_1 = K$ and compute a relation J as described by Alg. 1, regardless of the fact that K does not satisfy our requirements (intents of K need not be intents of I). The relation J will be a closed subrelation of I and it will contain K as a subset. Also note that the dual construction leads to a different closed subrelation.

Another open problem is whether it is possible to improve the estimation of the number of iterations of Alg. 1 from Prop. 9. In fact, we were not able to construct any example with the number of iterations greater than $\min(|X|, |Y|)$.

References

1. Bertet, K., Morvan, M.: Computing the sublattice of a lattice generated by a set of elements. In: Proceedings of Third International Conference on Orders, Algorithms and Applications. Montpellier, France (1999)
2. Ganter, B., Wille, R.: Formal Concept Analysis – Mathematical Foundations. Springer (1999)
3. Whitman, P.M.: Free lattices II. Annals of Mathematics 43(1), pp. 104–115 (1942)
4. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered Sets, pp. 445–470. Boston (1982)

RV-Xplorer: A Way to Navigate Lattice-Based Views over RDF Graphs

Mehwish Alam, Amedeo Napoli, and Matthieu Osmuk

LORIA (CNRS – Inria Nancy Grand Est – Université de Lorraine)
BP 239, Vandoeuvre-lès-Nancy, F-54506, France
{mehwish.alam,amedeo.napoli,matthieu.osmuk@loria.fr}

Abstract. More and more data are being published in the form of machine readable RDF graphs over Linked Open Data (LOD) Cloud accessible through SPARQL queries. This study provides interactive navigation of RDF graphs obtained by SPARQL queries using Formal Concept Analysis. With the help of this `View By` clause a concept lattice is created as an answer to the SPARQL query which can then be visualized and navigated using RV-Xplorer (Rdf View eXplorer). Accordingly, this paper discusses the support provided to the expert for answering certain questions through the navigation strategies provided by RV-Xplorer. Moreover, the paper also provides a comparison of existing state of the art approaches.

Keywords: RV-Xplorer, Lattice Navigation, SPARQL Query Views, Formal Concept Analysis

1 Introduction

Recently, Web Data is turning into "Web of Data" which contains the meta data about the web documents present in HTML and textual format. The goal behind this "Web of Data" is to make already existing data to be usable by not only human agents but also by machine agents. With the effort of Semantic Web community, an emerging source of meta data is published on-line called as Linked Open Data (LOD) in the form of RDF data graphs. There has been a huge explosion in LOD in recent past and is still growing. Up until 2014, LOD contains billions of triples. SPARQL[1] is the standard query language for accessing RDF graphs. It integrates several resources to generate the required answers For instance, queries such as *What are the movements of the artists displayed in Musee du Louvre?* can not be answered by standard search engines. Nowadays, Google has introduced a way of answering questions directly such as currency conversion, calculator etc. but such queries are answered based on most frequent queries posed by the experts.

When an expert poses a query to a search engine too many results are retrieved for the expert to navigate through, which may be cumbersome when a

[1] http://www.w3.org/TR/rdf-sparql-query/

expert has to go through a number of links to find the interesting ones, hence leading to the problem of *information overload* [3]. Same is the case with the answers obtained by SPARQL query with the `SELECT` [8]. Even if there are hundreds of answers, it becomes harder for the expert to find the interesting patterns. The current study is a continuation of Lattice-Based View Access (LBVA) [1] which provides a view over RDF graphs through SPARQL queries to give complete understanding of a part of RDF graph that expert wants to analyze with the help of Formal Concept Analysis. LBVA takes the SPARQL query and returns a concept lattice called as view instead of the results of the SPARQL query. These views created by LBVA are machine as well as human processable. Accordingly, RV-Xplorer (Rdf View eXplorer) exploits the powerful mathematical structure of these concept lattices thus making it interpretable by human. It also allows human agents to interact with the concept lattice and perform navigation. The expert can answer various questions while navigating the concept lattice. RV-Xplorer provides several ways to guide the expert during this navigation process.

This paper is structured as follows: section 2 gives the motivating example, section 3 introduces the required background knowledge to understand the rest of the paper. Section 4 details the elements of Graphical User Interface while section 5 and section 6 details the navigation operations as well as other functionalities supported by RV-Xplorer. Section 7 briefly discusses the related work. Finally, section 8 concludes the paper and discusses the future work.

2 Motivating Example

Consider a scenario where an expert wants to pose following questions based on articles published in conferences or journals from a team working on data mining. In the current study, we extract the papers published in "Orpailleur Team" in LORIA, Nancy, France. Following are the questions in which an expert may be interested in:

- What are the main research topics in the team and the key researchers w.r.t. these topics, for example, researchers involved in most of the papers in a prominent topic?
- What is the major area of the research of the leader of the team and various key persons?
- Can the diversity of the team leader and key persons be detected?
- Given a paper is it possible to retrieve similar papers published in the team?
- Who are the groups of persons working together?
- What are the research tendencies and possibly the forthcoming and new research topics (for example, single and recent topics which are not in the continuation of the present topics)?

Such kind of questions can not be answered by Google. In this paper we want to answer such kind of questions through lattice navigation supported by RV-Xplorer which is built from an initial query and then is explored by the expert according to her preferences.

3 Preliminaries

Linked Open Data: Linked Open Data (LOD) [2] is the way of publishing structured data in the form of RDF graphs. Given a set of URIs $\mathbf{U}$, blank nodes $\mathbf{B}$ and literals $\mathbf{L}$, an RDF triple is represented as $t = (s, p, o) \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{B} \cup \mathbf{L})$, where s is a subject, p is a predicate and o is an object. A finite set of RDF triples is called as RDF Graph $\mathcal{G}$ such that $\mathcal{G} = (V, E)$, where V is a set of vertices and E is a set of labeled edges. Each pair of vertices connected through a labeled edge keeps the information of a statement. Each statement is represented as $\langle subject, predicate, object \rangle$ referred to as an RDF Triple. V includes *subject* and *object* while E includes the *predicate*.

SPARQL: A standard query language for RDF graphs is SPARQL[2] which mainly focuses on graph matching. A SPARQL query is composed of two parts the head and the body. The body of the query contains the Basic Graph Patterns (present in the WHERE clause of the query). These graph patterns are matched against the RDF graph and the matched graph is retrieved and manipulated according to the conditions given in the query. The head of the query is an expression which indicates how the answers of the query should be constructed.

Let us consider a query from the scenario in section 2, Q = *Who is the team leader of the data mining team in loria*. For answering such questions consider an RDF resource containing all the papers ever published in the data mining team. With the help of SPARQL query the papers published in the last 5 years in English language can be extracted. The SPARQL representation of the query Q is shown in listing 1.1. Lines 1, 2 keep the information about the prefixes used in the rest of the query. Line 5, 6 and 7 retrieve all the papers with their authors and keywords. Line 8 and 9 retrieve the publication year of the paper and filter according to the condition.

```
1 PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX dc:<http://purl.org/dc/terms/>

3 SELECT distinct ?title ?keywords ?author
4 where {
5 ?paper dc:creator ?author .
6 ?paper dc:subject ?keywords .
7 ?paper dc:title ?title .
8 ?paper dcterms:issued ?publicationYear
9  FILTER(xsd:date(?publicationYear) >= '2011-01-01'^^xsd:date) }
```

Listing 1.1: SPARQL for extracting triples.

Lattice-Based View Access: Lattice-Based View Access [1], allows the classification of SPARQL query results into a concept lattice, referred to as a *view*, for data analysis, navigation, knowledge discovery and information retrieval purposes. It introduces a new clause `VIEW BY` which enhances the functionality of already existing `GROUP BY` clause in SPARQL query by adding sophisticated classification and Knowledge Discovery aspects.

[2] http://www.w3.org/TR/rdf-sparql-query/

The variable appearing in the `VIEW BY` clause of the SPARQL query is referred to as object variable[3] The rest of the variables are the attribute variables. Then the answer tuples obtained by the query are processed based on object and the attribute variables. The values obtained for the object variable are mapped to the objects in the formal context $\mathcal{K} = (G, M, I)$ and the answers obtained for attribute variables are mapped to the attributes in the context. Consider the query given in listing 1.1 with classification capabilities i.e., containing the clause `VIEW BY ?title` then the set of variables in the SELECT clause can be given as $V = \{$*?title, ?keyword, ?author*$\}$. The object variable will be *?title* and attribute variable will be *?keyword* and *?author*. After applying LBVA, the objects contain the titles of the paper and the attributes are the set of keywords and authors in the context. From this context, the concept lattice is built which is referred to as a *Lattice-Based View*.

LBVA is oriented towards the classification of SPARQL queries, but we can interpret the present research activity at a more general level, the classification of LOD. Accordingly, what is proposed in the paper is a tool for navigating a classification of LOD.

4 The RV-Xplorer

RV-Xplorer (Rdf View eXplorer) is a tool for navigating concept lattices generated by the answers of SPARQL queries over part of RDF graphs using Lattice-Based View Access. Accordingly, this tool provides navigation to the expert over the classification of SPARQL query answers for analyzing the data, finding hidden regularities and answering several questions. On each navigation step it guides the expert in decision making and performing selection to avoid unnecessary selections. It also allows the user to change her point of view while navigating i.e., navigation by extent. Moreover, it also allows the expert to only focus on the specific and interesting part of the concept lattice by allowing her to hide the part of lattice which is not interesting for her.

RV-Xplorer is a web-based tool for building concept lattices. On the client side it uses D3.js which stands for Data-Driven Documents and is based on Javascript for developing interactive data visualizations in modern web browsers. It also uses model-view-controller (MVC) which separates presentation, data and logical components. On the server side we use PHP and MySQL for computing and storing the data. Generally, data can be a graph or pattern generated by pattern mining algorithms etc. Currently, this tool is not publicly available.

Figure 1 shows the overall interface of RV-Xplorer (Rdf View eXplorer) which consists of three parts: (1) the middle part is called *local view* which shows detailed description of the selected concept allowing interaction, navigation and level-wise navigation, (2) the left panel is referred to as *Spy* showing the global view of the concept lattice and (3) the lower left is the summarization index for guiding the expert in making decision about which node to choose in the next

[3] The object here refers to the object in FCA.

level by showing the statistics of the next level. For the running scenario, the concept lattice is also available on-line[4].

4.1 Local View

Each selected node in the concept lattice is shown in the middle part of the interface displaying complete information. Let c be the selected concept such that $c \in C$ where C is the set of concepts in the complete lattice $\mathcal{L} = (C, \leq)$ then a local view shows the complete information about this concept i.e., the extent, intent and the links to the super-concept and the sub-concepts. The set of super and sub-concepts are linked to the selected node where each link represents the partially ordered relation $\leq$. By default, the top node is the selected node and is shown in *local view*.

Figure 1 (below) shows the selected concept, the orange part defines the label of the selected node which is the entry point for the concept, the pink and yellow parts give the labels of the super-concepts and sub-concepts connected to the selected concept respectively. The green and blue part give the information about the intent and the extent respectively.

4.2 Spy

A global view in left panel shows the map of the complete lattice $\mathcal{L} = (C, \leq)$ for a particular SPARQL query over an RDF Graph. It tracks the position of the expert in the concept lattice and the path followed by the expert to reach the current concept. It also helps in several navigation tasks such as direct navigation, changing navigation space and navigation between point-of-views. All of these navigation modes are discussed in section 5.

4.3 Statistics about the next level

The statistics about the next level are computed with the help of a summarization index which depicts the information about the distribution of the objects in the extent of the selected concept in the linked sub-concepts i.e., concepts in the next level of the concept lattice. Let c_i be a concept in the next level where $i \in \{1, \ldots, n\}$ and n is the number of concepts in the next level. $ext(c_i)$ is the extent of the concept then $|ext(c_i)|$ is the size of the extent. Finally, the statistics about the next level are computed with the help of summarization index.

$$summarization\ index = \frac{|ext(c_i)|}{\sum_{j=\{1,\ldots,n\}} |ext(c_j)|} \times 100 \tag{1}$$

Here, $\sum_{j=\{1,\ldots,n\}} |ext(c_j)|$ is the sum of extent size of all the concepts in the next level. The sum of summarization index for all the sub-concept adds to 100%. In Figure 1, the percentages are represented in the form of a pie-chart

[4] `http://rv-xplorer.loria.fr/#/graph/orpailleur_paper/1/`

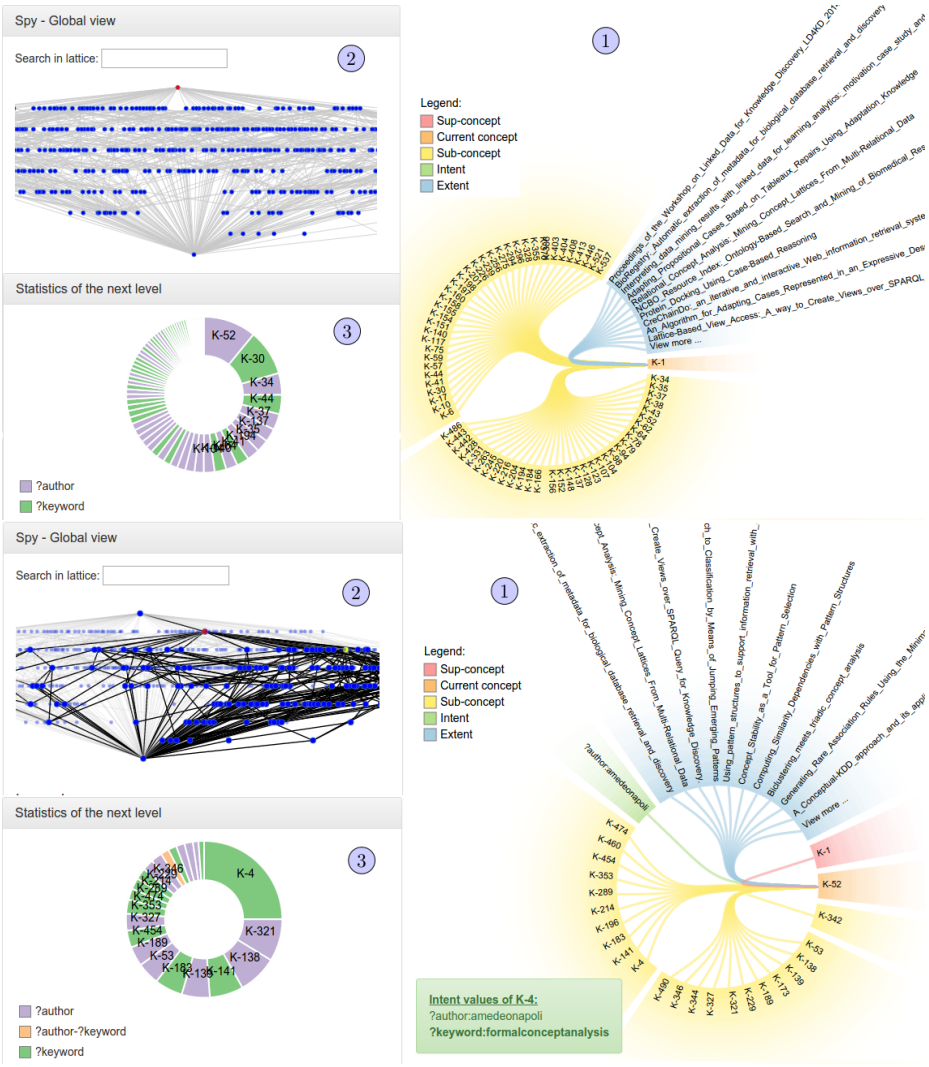


Fig. 1: Figure above shows the basic interface of RV-Xplorer displaying the top concept. The Figure below shows the local view of $K\#52$, the concept containing all the papers authored by Amedeo Napoli.

which shows the distribution. The sub-concept containing the most elements in the extent has the highest percentage and hence has the biggest part in the pie chart.

5 Navigation Operations

In this section we detail some of the classical [4] as well as advanced navigation operations that are implemented in RV-Xplorer. Navigation can be done locally with a parallel operation which is shown globally through local and global views. Navigation operations allow the expert to locate particular pieces of information which helps in obtaining several answers of the expert questions as well as analysis of the data at hand. Initially, the selected concept is the top concept which contains all the objects.

5.1 Guided Downward (Drill down)/ Upward Navigation (Roll-up):

The local view provides expert with the drilling down operation which is achieved by selecting the sub-concepts given in yellow part of local view. RV-Xplorer guides the expert in drilling down the concept lattice by showing contents of the sub-concept to the expert before selecting the node on mouse over. Another added guidance provided to the expert is with the help of the summarization index which gives the statistics about the next level. This way the expert can avoid the attributes or the navigation path which may lead to uninteresting results. The local view also allows the expert to roll-up from the specific concept to the general concept. A super-concept can be selected following the link given in the view.

Consider the running scenario discussed in section 2 where the expert wants to know *who are researchers having main influences in the team?* by analyzing the publications of this particular team. Initially, the selected concept in the local view is the top concept (see Figure 1 (above)). Now it can be seen from the summarization index that most of the papers are contained in $K\#52$. On mouse over on $K\#52$ it shows that this concept keeps all the papers published by Amedeo Napoli. From here it can be safely concluded that Amedeo Napoli is the leader of the team. Similarly, several key team members can be identified on the same level such as supervisors etc. If the expert wants to view the papers published by Amedeo Napoli, a downward navigation is performed by selecting concept $K\#52$. With the help of the summarization index another question can be answered i.e., *what are the main research topics of these researchers?*. Again by consulting the index it can be seen that $K\#4$ keeps the largest percentage of papers published by Amedeo Napoli (see Figure 1 (below)) and the keyword in this concept is Formal Concept Analysis meaning that the main area of research of Amedeo Napoli is Formal Concept Analysis. However, there are many other areas of research on which he has worked, which shows the diversity of authors based on the area of research he has published in. Moreover, the sub-lattice connected to this concept keeps information about the community of authors

with who she publishes the most and about which topic and what variants of formal concept analysis. Now, if the expert wants to retrieve all the papers published by Amedeo Napoli then she can go back to $K\#52$.

5.2 Direct Navigation

The spy on the left part of the RV-Xplorer (see Figure 1) allows the expert for direct navigation. If an expert has navigated too deep in the view while performing multiple drill-down operations then the spy, which keeps track of the current position of the expert, shows all the paths from the selected concept to the top concept and allows the expert to directly jump from one concept to another linked concept without performing level-wise navigation. Unlike drill-down and roll-up, direct navigation allows the expert to skip two or more hops and select the more general or specific concept.

These three navigation modes are very common and are repeatedly discussed in many of the navigational tools built for concept lattice such as Camelis [11] and CREDO [5] which may or not may not be for a specific purpose. The main difference between RV-Xplorer and the two approaches and most of the navigational tools is that they use folder-tree display. As a contrast we manage to keep the original structure of a concept lattice. An added advantage of RV-Xplorer is that these navigation modes are guided at each step meaning that the interface shows the expert with what is contained in the next node as well as the statistics about the next level. This way the interface guides the expert in choosing the nodes interesting for her by reducing the chance of performing unnecessary navigation and backtracking to see the details unnecessarily.

5.3 Navigating Across Point-of-Views

The current interface allows the expert to toggle between points-of-view, i.e., at any point an expert can start exploring the lattice with respect to the objects (extent) in the concept lattice. Let c be the selected concept and the expert is interested in $g_1 \in ext(c)$ where $ext(c)$ is the extent of the selected concept. Then if the expert hovers her mouse over this extent in the local view, the Spy highlights all the concepts where this object is present along with the object concept of g_1 which is highlighted in red.

For instance, the selected concept contains keyword *data dependencies* in the intent and she is interested in the paper *Computing Similarity Dependencies with Pattern Structures* and she wants to retrieve all the related or similar papers then on mouse hover it highlights all the concepts containing this paper. Then she selects the concept highlighted in red i.e., the object concept of this paper. The right side of Figure 2 shows the highlighted object concept of *Computing Similarity Dependencies with Pattern Structures* in RV-Xplorer. After this concept is selected. The spy highlights all the paths from this concept until bottom and the top which actually is the sub-lattice associated to this paper. All the objects contained in the extent of the concepts in this sub-lattice are similar to

the paper at hand i.e., papers sharing some properties with the paper *Computing Similarity Dependencies with Pattern Structures*.

If we consider the folder-tree display as discussed in most of the navigational tools such as Camelis [11], CREDO [5] and CEM [7], such kind of navigation is not possible because it only allows navigation w.r.t. intent and extent is considered as the answers of the navigations. In case of RV-Xplorer, it is possible to obtain the sub-lattice related to a certain interesting object and this way the whole sub-lattice connected to the object concept of the object of interest can be navigated to retrieve similar objects i.e., sharing at least one attribute with the object of interest.

5.4 Altering Navigation Space

The navigation space can be changed when the selected concept is deep-down in the concept lattice without the effort to start the navigation all over again from the top concept. Let c be the selected concept such that m_1 *and* $m_2 \in int(c)$ (int(c) is the intent of the selected concept) and the expert has navigated downwards from the concept whose intent only contains m_1. Now the expert wants to navigate the lattice w.r.t. m_2, on mouse hover the interface highlights all the concepts where the given attribute exists and further highlights the attribute concept in red. The attribute concept of m_2 can be selected. In the running example, if the expert has navigated the lattice w.r.t. the author Amedeo Napoli and she finds some papers on FCA authored by Amedeo Napoli. Now she wants to navigate the concept lattice w.r.t. the keyword *FCA* then she can easily locate the attribute concept of the keyword FCA and navigate to get specific information. The left side of Figure 2 shows the highlighted attribute concept of FCA in RV-Xplorer.

In tree-folder display altering navigation space w.r.t. intent needs the expert to locate the attribute concept by herself by manually checking each of the branches because it represents the concept lattice as a tree. The problem with such a display is that it is not easy to alter the browsing space quickly or change the navigation point of view. Moreover, the sub-lattice connected to a selected concept can not be seen because of the restrictions posed by tree display.

5.5 Area Expansion

Area expansion allows the expert to select several concepts at one time scattered over the concept lattice and gives the overall view of what these concepts contains. These concepts are not necessarily a part of navigation path that the expert is following. It allows the expert to have an overall view of other concepts without starting the navigation process again.

This idea was first put-forth in [14], where they allow the expert to move from one concept lattice to another concept lattice based on the granularity level w.r.t. a taxonomy and a similarity threshold. The concepts in the concept lattice with higher threshold contains more detailed information as compared to the concept lattice built using lesser threshold. One drawback of such kind

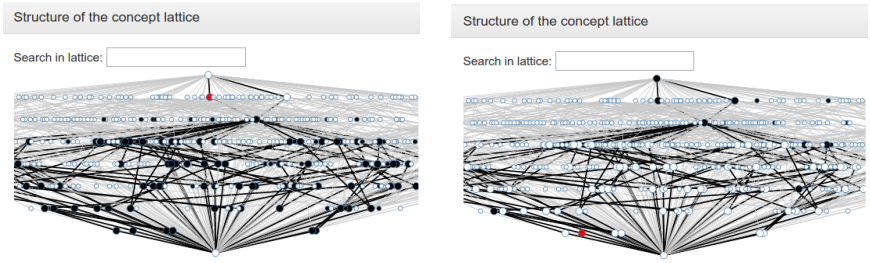


Fig. 2: Left Figure shows the Attribute Concept of FCA and Right Figure shows the Object Concept of *Computing Similarity Dependencies with Pattern Structures*.

of zooming operation is that it requires the computation of several concept lattices. In case of RV-Xplorer, we are dealing with simple concept lattice instead of the one created after using hierarchies meaning that all such kind of information needs to be scaled to obtain a binary context. As we are dealing with concept lattices built from binary contexts, we bend this functionality to suit the needs. It does not require computation of many concept lattices as well as no re-computation is required.

6 Hiding Non-Interesting Parts of the View

One of the most interesting characteristic of RV-Xplorer is that it allows the expert to hide the non-interesting part of the lattice. Let us consider that expert selects a concept c and it contains an attribute which is not interesting for her. She can at any point right click on the concept and select hide sub-lattice. One of the most interesting characteristic of a concept lattice is that if one concept contains some attribute in an intent then all the sub-concepts inherit this attribute. This way if the expert considers one concept as un-interesting then the whole sub-lattice will be considered as uninteresting and hence will be hidden from the expert while navigation. Such kind of functionality enables expert to reduce her navigational space and at the end the concept lattice contains only those concepts which are interesting for the expert.

Similar functionality was first introduced in CreChainDo system [15]. Similar to CREDO [5], CreChainDo allows the expert to pose a query against the standard search engine which returns some results. These results are then organized in the form of a concept lattice and displayed to the expert in the form of folder-tree display. An added advantage of CreChainDo over CREDO is that the former allows expert interaction i.e., the expert can mark the concepts as relevant or irrelevant based on her priorities. After the expert has marked the concept irrelevant the sub-lattice linked to that concept is deleted. Meaning that, it reduces the context based on this feedback and the concept lattice is computed again using the reduced context. In case of RV-Xplorer, the concept lattice is built on top of RDF graphs. Moreover, we do not recompute the lattice or remove anything from the concept lattice. We only hide the non-interesting part

of the lattice to reduce the navigation space of the expert. This way a reduction in the navigation space is performed without re-computing a concept lattice.

7 Related Tools

There have already been many efforts for providing expert the facilities to interact with the concept lattice applied to different domains. In [13], the authors discuss a query-based faceted search for Semantic Web, as a contrast we are mostly dealing with navigational capabilities that can be provided by utilizing the powerful structure introduced by Hasse Diagram. [10] proposes another interesting way of navigating the concept lattice which allows the novice user to navigate through the concept lattice without having to know the structure of the concept lattice. Same is the case with SPARKLIS [12], where user can perform selections and the tool acts as a query builder. As a contrast, RV-Xplorer provides exploration/navigational capabilities over SPARQL query answers with the help of view i.e., a concept lattice for data analysis and information retrieval purposes. Conexp[5] is another tool for visualizing small lattices. As a contrast, RV-Xplorer allows area expansion and also provides guided navigation. [1] discusses that the views generated are easily navigable by machine as well as human agents. Machine agents may access the datasets through SPARQL queries for application development purposes through generic SPARQL queries generating huge number of answers and consequently large number of concepts are provided by `View By` clause. However, when human agents want to access the information through SPARQL query they run specialized queries which do not generate huge number of answers. In the current study we are focusing on manageable number of answers to be visualized by human agents using our visualization software.

An added advantage over these approaches is that RV-Xplorer provides guidance to the expert at each step for making the decision about concept selection. This guidance is provided by showing the user at each step, the contents of the intent of next level, by showing the distribution of the extent with the help of summarization index and finally with the help of global view many other ways of guidance are provided.

8 Discussion

In this study we introduce a new navigational tool for concept lattices called as RV-Xplorer which provides exploration over SPARQL query answers. With the help of guided navigation implemented in RV-Xplorer we were able to answer all the questions posed initially in the scenario. However, this tool is not designed for only specific purpose any kind of concept lattice can be visualized and data from any domain can be analyzed using this tool. The RV-Xplorer tool is still in development and other functionalities should be added such as incremental visualization (w.r.t. a set of given objects and attributes), iceberg visualization

[5] `http://conexp.sourceforge.net/`

(given a set of attributes and objects, and a frequency threshold), integration of quality measures, visualization of implications and Duquenne-Guigues basis... We believe that visualization tools, as many other researchers do (see the tools discussed in [6]) are of main importance, not only for FCA but for data mining in general. Accordingly, a new generation of visualization tools should be studied and designed, and RV-Xplorer is an example of this new tools and what can be imagined for supporting the analyst in the mining activity. We also want to perform human evaluation of the tool as discussed in [10] and [13].

References

1. Mehwish Alam and Amedeo Napoli. Defining views with formal concept analysis for understanding SPARQL query results. In *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications.*, 2014.
2. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
3. Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38, 2009.
4. Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122, 1996.
5. Claudio Carpineto and Giovanni Romano. Exploiting the potential of concept lattices for information retrieval with CREDO. *J. UCS*, 10(8):985–1013, 2004.
6. Víctor Codocedo and Amedeo Napoli. Formal concept analysis and information retrieval - A survey. In *Formal Concept Analysis - 13th International Conference, ICFCA 2015, Nerja, Spain, June 23-26, 2015, Proceedings*, pages 61–77, 2015.
7. Richard Cole and Gerd Stumme. CEM - A conceptual email manager. In *8th International Conference on Conceptual Structures, ICCS 2000, Darmstadt, Germany, August 14-18, 2000, Proceedings*, pages 438–452, 2000.
8. Claudia d'Amato, Nicola Fanizzi, and Agnieszka Lawrynowicz. Categorize by: Deductive aggregation of semantic web query results. In *ESWC (1)*, 2010.
9. Peter W. Eklund, editor. *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*, Lecture Notes in Computer Science. Springer, 2004.
10. Peter W. Eklund, Jon Ducrou, and Peter Brawn. Concept lattices for information visualization: Can novices read line-diagrams? In Eklund [9], pages 57–73.
11. Sébastien Ferré. Camelis: a logical information system to organise and browse a collection of documents. *Int. J. General Systems*, 38(4):379–403, 2009.
12. Sébastien Ferré. Expressive and scalable query-based faceted search over SPARQL endpoints. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, 2014.
13. Sébastien Ferré and Alice Hermann. Reconciling faceted search and query languages for the semantic web. *IJMSO*, 7(1):37–54, 2012.
14. Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smaïl-Tabbone. Using domain knowledge to guide lattice-based complex data exploration. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, pages 847–852, 2010.
15. Emmanuel Nauer and Yannick Toussaint. Dynamical modification of context for an iterative and interactive information retrieval process on the web. In *Proceedings of the Fifth International Conference on Concept Lattices and Their Applications, CLA 2007, Montpellier, France, October 24-26, 2007*, 2007.

Finding p-indecomposable Functions: FCA Approach

Artem Revenko[1,2]

[1] TU Wien
Karlsplatz 13, 1040 Vienna, Austria
[2] TU Dresden
Zellescher Weg 12-14, 01069 Dresden, Germany

Abstract. The parametric expressibility of functions is a generalization of the expressibility via composition. All parametrically closed classes of functions (p-clones) form a lattice. For finite domains the lattice is shown to be finite, however straight-forward iteration over all functions is infeasible, and so far the p-indecomposable functions are only known for domains with two and three elements. In this work we show how p-indecomposable functions can be computed more efficiently by means of an extended version of attribute exploration (AE). Due to the growing number of attributes standard AE is not able to guarantee the discovery of all p-indecomposable functions. We introduce an extension of AE and investigate its properties. We investigate the conditions allowing us to guarantee the success of exploration. In experiments the lattice of p-clones on three-valued domain was reconstructed.

Keywords: parametric expressibility, attribute exploration, p-indecomposable function

1 Introduction

The expressibility of functions is a major topic in mathematics and has a long history of investigation. The interest is explainable: when one aims at investigating any kind of functional properties, which classes of functions should one consider? If a function f is expressible through a function h then it often means that f inherits properties of h and should not be treated separately. Moreover, if h in turn is expressible through f then both have similar or even the same properties. Therefore, partition with respect to expressibility is meaningful and can be the first step in the investigation of functions.

With the development of electronics and logical circuits a new question arises: if one wants to be able to express all possible functions which minimal set of functions should one have at hands? One of the first investigations in this direction was carried out in [Pos42]; in this work all the Boolean classes of functions closed under expressibility are found and described. Afterwards many important works were dedicated to related problems such as the investigation of the structure of the lattice of functional classes, for example, [Yab60,Ros70]. However, it

is known that the lattice of classes of functions closed under expressibility is in general uncountably infinite. In [Kuz79] a more general type of functional expressibility was introduced – parametric expressibility. A significant advantage of this type of expressibility is that for any finite domain $A_k, |A|= k$ the lattice of all classes closed under parametric expressibility classes of functions (p-clones) is finite [BW87]. However, finding this lattice is a complex task. For $k = 3$ in a thorough and tedious investigation [Dan77] it was proved that a system of 197 functions forms the lattice of all p-clones. The investigation was carried out without the use of computers.

In this paper we introduce, develop, and investigate the methods and tools for automation of the exploration of the lattice of p-clones. Therefore, this paper "applied" to A_3 can be seen as complementing the work [Dan77] where a proof of the correctness of the results obtained using the elaborated in this paper tools can be found. Namely, in this paper we answer the question **how** to find all the p-clones, whereas in [Dan77] it is proved that certain functions allow us to construct the desired lattice. The presented methods and tools are extensible to larger domains as well.

Contributions

- New original approach to exploring the lattice of p-clones introduced;
- An extension of the standard exploration procedure is introduced and investigated;
- The whole procedure is implemented and executed; the obtained results confirm with the previously known results;
- It is proved that for certain starting conditions the desired lattice will necessarily be eventually discovered.

2 Formal Concept Analysis

In what follows we keep to standard definitions of FCA [GW99]. Let G and M be sets and let $I \subseteq G \times M$ be a binary relation between G and M. The triple $\mathbb{K} := (G, M, I)$ is called a *(formal) context*. The set G is called the set of *objects*. The set M is called the set of *attributes*. A context (G_*, M_*, I_*) such that $G_* \subseteq G,\ M_* \subseteq M$, and $I_* = I \cap G_* \times M_*$ is called a *subcontext* of $\mathbb{K}$.

Consider mappings $\varphi\colon 2^G \to 2^M$ and $\psi\colon 2^M \to 2^G$:

$$\varphi(X) := \{m \in M \mid gIm \ \text{ for all } g \in X\},$$

$$\psi(A) := \{g \in G \mid gIm \ \text{ for all } m \in A\}.$$

Mappings φ and ψ define a *Galois connection* between $(2^G, \subseteq)$ and $(2^M, \subseteq)$, i.e. $\varphi(X) \subseteq A \Leftrightarrow \psi(A) \subseteq X$. Usually, instead of φ and ψ a single notation $(\cdot)'$ is used.

Let $X \subseteq G,\ A \subseteq M$. A *formal concept* C of a formal context (G, M, I) is a pair (X, A) such that $X' = A$ and $A' = X$. The subset of objects X is called the

extent of C and is denoted by $\mathrm{ext}(C)$, and the subset of attributes A is called the *intent of* C and is denoted by $\mathrm{int}(C)$. For a context (G, M, I), a concept $C_1 = (X, A)$ is a *subconcept* of a concept $C_2 = (Y, B)$ ($C_1 \leq C_2$) if $X \subseteq Y$ or, equivalently, $B \subseteq A$. This defines a partial order on formal concepts. The set of all formal concepts of (G, M, I) is denoted by $\mathfrak{B}(G, M, I)$.

An *implication* of $\mathbb{K} = (G, M, I)$ is defined as a pair (A, B), where $A, B \subseteq M$, written $A \to B$. A is called the *premise*, B is called the *conclusion* of the implication $A \to B$. The implication $A \to B$ is *respected by a set of attributes* N if $A \nsubseteq N$ or $B \subseteq N$. We say that the implication is *respected by an object* g if it is respected by the intent of g. If g does not respect an implication then g is called a *counter-example*. The implication $A \to B$ *holds* (is *valid*) in $\mathbb{K}$ if it is respected by all g', $g \in G$, i.e. every object, that has all the attributes from A, also has all the attributes from B ($A' \subseteq B'$). A *unit implication* is defined as an implication with only one attribute in its conclusion, i.e. $A \to b$, where $A \subseteq M$, $b \in M$. Every implication $A \to B$ can be regarded as a set of unit implications $\{A \to b \mid b \in B\}$.

An *implication basis* of a context $\mathbb{K}$ is defined as a set $\mathfrak{L}_{\mathbb{K}}$ of implications of $\mathbb{K}$, from which any valid implication for $\mathbb{K}$ can be obtained as a consequence and none of the proper subsets of $\mathfrak{L}_{\mathbb{K}}$ has this property. We call the set of all valid in $\mathbb{K}$ the *implicative theory* of $\mathbb{K}$. A minimal in the number of implications basis was defined in [GD86] and is known as the *canonical implication basis*.

An object g is called *reducible* in a context $\mathbb{K} := (G, M, I)$ iff $\exists X \subseteq G \setminus g : g' = X'$. Note that a new object is going to be reducible if in the context there already exists a formal concept with the same intent as the intent of the new object. Reducible objects neither contribute to any implication basis nor to the concept lattice [GW99], therefore, if one is only interested in the implicative theory or in the concept lattice of the context reducible objects can be eliminated. In what follows we introduce other types of reducibility, therefore, we refer to this type of reducibility as *plain* reducibility.

In what follows the canonical implication basis is used, however, the investigation could be performed using another implication basis.

Attribute Exploration (AE) consists in iterations of the following steps until stabilization: computing the implication basis of a context, finding counterexamples to implications, updating the context with counterexamples as new objects, recomputing the basis. AE has been successfully used for investigations in many mostly analytical areas of research. For example, in [KPR06] AE is used for studying Boolean algebras, in [Dau00] lattice properties are studied, in [Rev14] algebraic identities are studied.

3 Expressibility of Functions

Consider a set A_k, $|A| = k, k \in \mathbb{N}$. Consider a function $f : A^{ar(f)} \to A$ ($ar(f)$ denotes the arity of f), the set of all possible functions over A_k of different arities is denoted by U_k. The particular functions $p_n^i(x_1, \ldots x_n) = x_i$ are called

the *projections*. The set of all projections is denoted by Pr. In what follows instead of writing $(x_1, \dots x_n)$ we use a shorter notation $(\mathbf{x})$.

Let $H \subseteq U_k$. We say that f is *compositionally expressible* through H (denoted $f \leq H$) if the following condition holds:

$$f(\mathbf{x}) \quad \equiv \quad h(j_1(\mathbf{x}), \dots, j_{ar(h)}(\mathbf{x})), \tag{1}$$

for some $h, j_1, \dots j_m \in H \cup Pr$.

A *functional clone* is a set of functions containing all projections and closed under compositions. The set of all functional clones over a domain of size $k = 2$ forms a countably infinite lattice [Pos42]. However, if $k > 2$ then the set of all functional classes is uncountable [YM59].

Let $H \subseteq U_k$ and for any $i \in [1, m] : \ t_i, s_i \in H \cup Pr$. We say that $f, \ f \in U_k$ is *parametrically expressible* through H (denoted $f \leq_p H$) if the following condition holds:

$$f(\mathbf{x}) = y \quad \Longleftrightarrow \quad \exists \mathbf{w} \bigwedge_{i=1}^{m} t_i(\mathbf{x}, \mathbf{w}, y) = s_i(\mathbf{x}, \mathbf{w}, y). \tag{2}$$

The notation $J \leq_p H$ means that every function from J is parametrically expressible through H. A *parametric clone* (or *p-clone*) is a set of functions closed under parametric expressibility and containing all projections. We consider a special relation $f^\bullet$ of arity $ar(f)+1$ on A_k called the *graph* of function f. $f^\bullet$ consists of the tuples of the form $(\mathbf{x}, f(\mathbf{x}))$. If function h is compatible with $f^\bullet$, i.e. if for all valuations of variables x_{ij} in A_k holds the identity $(ar(f) = n, \ ar(h) = m)$

$$f(h(x_{11}, \dots, x_{1m}), \dots h(x_{n1}, \dots, x_{nm})) \equiv h(f(x_{11}, \dots, x_{n1}), \dots f(x_{1m}, \dots, x_{nm})),$$

then we say that functions f and h *commute* (denoted $f \perp h$). For a set of functions H we write $f \perp H$ to denote that for all $h \in H : \ f \perp h$. The commutation property is commutative, i.e. $f \perp h$ iff $h \perp f$.

The *centralizer* of H is defined by $H^\perp = \{g \in U_k \mid g \perp H\}$. In [Kuz79] it is shown that if $f \leq_p H$ then $f \perp H^\perp$.

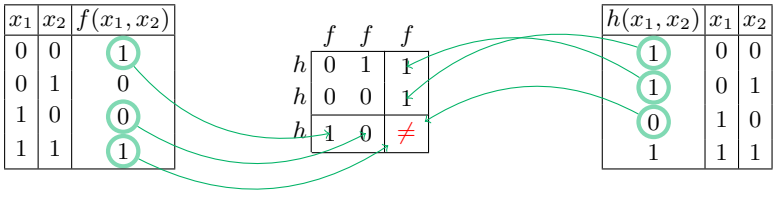


Fig. 1. Functions f and h do not commute

A function f is called *p-indecomposable* if each system H parametrically equivalent to $\{f\}$ (i.e. $f \leq_p H$ and $H \leq_p f$) contains a function parametrically equivalent to f. Hence, for each p-indecomposable function there exists a class of

p-indecomposable functions that are parametrically equivalent to it. From each such class we take only one representative (only one p-indecomposable function) and gather them in a set of p-indecomposable functions denoted by F_k^p. A p-clone H cannot be represented as an intersection of p-clones strictly containing H if and only if there exists a p-indecomposable function f such that $H = f^{\perp\perp}$. Hence, in order to construct the lattice of all p-clones it suffices to find all p-indecomposable functions. The lattice of all p-clones for any finite k is finite [BW87], hence, F_k^p is finite.

In [BW87] it is proved that it suffices to consider p-indecomposable functions of arity at most k^k, however, the authors conjecture that the actual arity should be equal to k for $k \geq 3$. The conjecture is still open. Nevertheless, thanks to results reported in [Dan77], we know that the conjecture holds for $k = 3$.

4 Exploration of P-clones

The knowledge about the commutation properties of a finite set of functions $F \subseteq U_k$ can be represented as a formal context $\mathbb{K}_F = (F, F, \perp_F)$, where $\perp_F \subseteq F^2$, a pair $(f_1, f_2) \in F^2$ belongs to the relation $\perp_F$ iff $f_1 \perp f_2$. Note that the relation $\perp_F$ is symmetric, hence, the objects and the attributes of the context are the same functions.

The goal of this paper is to develop methods for constructing the lattice of all p-clones on A_3. As already noted, for the purpose of constructing the lattice of p-clones it suffices to find all p-indecomposable functions F_k^p. The set of supremum-irreducible elements of the lattice of p-clones is exactly the set $\{f^{**} \mid f \in F_k^p\}$.

For any domain of size k there exist k^{k^k} functions of arity k. Therefore, to compute the context of all commuting functions $\mathbb{K}_{U_k}$ one has to perform $O(k^{k^k} * k^{k^k} * k^{k^2})$ operations (taking into consideration only functions of arity k and the cost of commutation check in the worst case). For $k = 3$ we count about 10^{30} operations. Therefore, already for $k = 3$ a brute-force solution is infeasible.[3]

We intend to apply AE to commuting functions. For this purpose we developed and implemented methods for finding counter-examples to implications over functions from U_k [Rev15]. These methods are not presented in this paper for the sake of compactness. However, as the number of attributes is not fixed, the success of applying AE is not guaranteed, i.e. it is not guaranteed that the complete lattice of p-clones will eventually be discovered using AE.

4.1 Object-Attribute Exploration

We now describe which commuting properties a new function $g \notin F$ should possess in order to alter the concept lattice of the original context $\mathbb{K} = (F, F, \perp)$ despite the fact that the intent of g is equal to an intent from $\mathfrak{B}(F, F, \perp_F)$.

[3] Of course one can use dualities, but it does not give a feasible solution as well as there exist only $k * (k - 1)$ dualities.

To distinguish between binary relations on different sets of functions we use subscripts. The commutation relation on F is denoted by $\perp_F$, i.e. $\perp_F = \{(h, j) \in F^2 \mid h \perp j\}$. The context with the new function $(F \cup g, F \cup g, \perp_{F \cup g})$ is denoted by $\mathbb{K}_{F \cup g}$. The derivation operator for the context $\mathbb{K}_{F \cup g}$ is denoted by $(\cdot)^{\perp_{F \cup g}}$.

Proposition 1. *Let $C \in \mathfrak{B}(F, F, \perp)$ such that ext$(C) \not\subseteq$ int(C). Let $g \in U_k, g \notin F$ be a function such that $g^{\perp_{F \cup g}} \cap F =$ int(C) (g is reducible in $\mathbb{K}_F$).*
g is irreducible in $\mathbb{K}_{F \cup g}$ $\quad \Leftrightarrow \quad$ $g \perp g$.

Proof. As $\text{ext}(C) \not\subseteq \text{int}(C)$ and for all $f \in F \setminus \text{int}(C) : \ g \not\perp f$ it follows that $g \not\perp \text{ext}(C)$. We prove the contrapositive statement: g is reducible in $\mathbb{K}_{F \cup g}$ $\quad \Leftrightarrow$ $g \not\perp g$.

$\Leftarrow$ As $g \not\perp g$ we have $g^{\perp_{F \cup g}} = \text{int}(C) = \text{ext}(C)^{\perp_{F \cup g}}$. Therefore, g is reducible.
$\Rightarrow$ As g is reducible we obtain $g^{\perp_{F \cup g}} = H^{\perp_{F \cup g}}$ for some $H \subseteq F$. Fix this H. As $H^{\perp_{F \cup g}} = \text{int}(C)$ we have $H^{\perp_{F \cup g} \perp_{F \cup g}} = \text{ext}(C)$. Suppose $H \subseteq \text{int}(C)$, then $H^{\perp_{F \cup g} \perp_{F \cup g}} \subseteq \text{int}(C)^{\perp_{F \cup g} \perp_{F \cup g}} = \text{int}(C)$. As $H^{\perp_{F \cup g} \perp_{F \cup g}} = \text{ext}(C)$ and $\text{ext}(C) \not\subseteq \text{int}(C)$ we arrive at a contradiction. Therefore, $H \not\subseteq \text{int}(C)$. Hence, $g \not\perp H$, therefore, $g \notin H^{\perp_{F \cup g}}$, hence, $g \notin g^{\perp_{F \cup g}}$.

Corollary 1. *If g is reducible in $\mathbb{K}_F$, but irreducible in $\mathbb{K}_{F \cup g}$ and $g \perp g$ then ext$(C) \to g$ holds in $\mathbb{K}_{F \cup g}$.*

Proof. As $g^{\perp_{F \cup g}} = \text{int}(C) \cup \{g\}$ and $\text{ext}(C)^{\perp_{F \cup g}} = \text{int}(C)$ we have $\text{ext}(C)^{\perp_{F \cup g}} \subset g^{\perp_{F \cup g}}$, therefore, $\text{ext}(C) \to g$.

The statement dual to Proposition 1 holds as well.

Proposition 2. *Let $C \in \mathfrak{B}(F, F, \perp_F)$ such that ext$(C) \subseteq$ int(C). Let $g \in U_k, g \notin F$ be a function such that $g^{\perp_{F \cup g}} \cap F =$ int(C) (g is reducible in $\mathbb{K}_F$).*
g is irreducible in $\mathbb{K}_{F \cup g}$ $\quad \Leftrightarrow \quad$ $g \not\perp g$.

Proof. As $\text{ext}(C) \subseteq \text{int}(C)$ and $g \perp \text{int}(C)$ then $g \perp \text{ext}(C)$. We prove the contrapositive statement: g is reducible in $\mathbb{K}_{F \cup g}$ $\quad \Leftrightarrow \quad$ $g \perp g$.

$\Leftarrow$ As $g \perp g$ and $g \perp \text{ext}(C)$ we have $\text{ext}(C)^{\perp_{F \cup g}} = \text{int}(C) \cup \{g\} = g^{\perp_{F \cup g}}$. Hence, g is reducible.
$\Rightarrow$ As g is reducible we obtain $g^{\perp_{F \cup g}} = H^{\perp_{F \cup g}}$ for some $H \subseteq F$. Fix this H. As $g \perp \text{int}(C)$ we have $H \perp \text{int}(C)$, hence, $H \subseteq \text{ext}(C)$. As $g \perp \text{ext}(C)$ we have $g \perp H$, hence, $g \in H^{\perp_{F \cup g}}$, therefore, $g \in g^{\perp_{F \cup g}}$ and $g \perp g$.

Corollary 2. *If g is reducible in $\mathbb{K}_F$, but irreducible in $\mathbb{K}_{F \cup g}$ and $g \not\perp g$ then $g \to$ ext(C) holds in $\mathbb{K}_{F \cup g}$.*

Proof. As $g^{\perp_{F \cup g}} = \text{int}(C)$ and $\text{ext}(C)^{\perp_{F \cup g}} = \text{int}(C) \cup \{g\}$ we have $g^{\perp_{F \cup g}} \subset \text{ext}(C)^{\perp_{F \cup g}}$, therefore, $g \to \text{ext}(C)$.

In order to distinguish reducibility in the old context $\mathbb{K}_F$ and in the new context $\mathbb{K}_{F \cup g}$ we introduce a new notation.

Definition 1. *We call a function g that is reducible in $\mathbb{K}_F$, but irreducible in $\mathbb{K}_{F\cup g}$,* first-order irreducible for $\mathbb{K}_F$. *If g is reducible for $\mathbb{K}_F$ and reducible in $\mathbb{K}_{F\cup g}$ we call it* first-order reducible for $\mathbb{K}_F$.

We remind that if g is irreducible in $(F\cup g, F, \perp_F \cup\{(g,f) \in \{g\}\times F \mid f \perp g\})$ we call it plainly irreducible. Hence, if function is first-order reducible for $\mathbb{K}_F$ then it is also plainly reducible in $\mathbb{K}_F$. Note that g is plainly irreducible in $\mathbb{K}_F$ iff g is a counter-example to some valid in $\mathbb{K}_F$ implication.

Next we present an example with functions from U_3, in order to explicitly show this we add 3 in the subscript of every function. The numbering of the functions is induced by the lexicographic ordering on the outputs of the functions [Rev15]. We use superscripts $\cdot^u$ for unary, $\cdot^b$ for binary, and $\cdot^t$ for ternary functions.

Example 1. The context under consideration $\mathbb{K}_0^{(3)}$ is presented in Figure 2. The implication basis of $\mathbb{K}_0^{(3)}$ is empty, therefore, there exist no plainly irreducible functions. The function $f^b_{3,756}$ has the following commuting properties: $f^b_{3,756} \perp \{f^u_{3,0}, f^b_{3,12015}\}$ and $f^b_{3,756} \not\perp f^u_{3,1}$. Moreover, $f^b_{3,756} \not\perp f^b_{3,756}$ and for the corresponding concept C holds $\mathrm{ext}(C) = \{f^u_{3,0}\} \subset \{f^u_{3,0}, f^b_{3,12015}\} = \mathrm{int}(C)$. As follows from Proposition 2, the function $f^b_{3,756}$ is first-order irreducible for $\mathbb{K}_0^{(3)}$.

	$f^u_{3,0}$	$f^u_{3,1}$	$f^b_{3,12015}$
$f^u_{3,0}$	×		×
$f^u_{3,1}$		×	×
$f^b_{3,12015}$	×	×	

Fig. 2. Context $\mathbb{K}_0^{(3)}$ of functions on domain A_3 containing $f^u_{3,0}, f^u_{3,1}, f^b_{3,12015}$

Corollary 3. *Let $C \in \mathfrak{B}(F, F, \perp_F)$, $g \in U_k, g \notin F$, and g be first-order reducible for $\mathbb{K}_F$.*

$$ext(C) \perp g \quad \Leftrightarrow \quad g \perp g.$$

Proof. Follows from Propositions 1 and 2 and the fact that $\mathrm{ext}(C) \perp g \Leftrightarrow \mathrm{ext}(C) \subseteq \mathrm{int}(C)$.

There remains a possibility that a union of sets of reducible functions is irreducible. We proceed with the simplest case when there are only two sets each containing a single first-order reducible function for the current context. We prove several propositions about such pairs of first-order reducible functions. The consequences of these propositions are deeper investigated in Section 4.2.

We consider a context $\mathbb{K}_F$ and new functions $g_1, g_2 \in U_k,\ g_1, g_2 \notin F$. We denote $\{g_1, g_2\}$ by G, $\perp_{F\cup G} = \{(h,j) \in (F\cup G)^2 \mid h \perp j\}$, the context $(F \cup$

$G, F \cup G, \perp_{F\cup G})$ is denoted by $\mathbb{K}_{F\cup G}$, the corresponding derivation operator is denoted by $(\cdot)^{\perp_{F\cup G}}$. As in the case with one function, for $i \in \{1, 2\} : g_i$ is not a counter-examples to a valid implication iff $g_i^{\perp_{F\cup G}} \cap F \in \text{int}(G, M, I)$. We denote the corresponding intents by $\text{int}(C_1)$ and $\text{int}(C_2)$, respectively.

Proposition 3. *Let $C_1, C_2 \in \mathfrak{B}(F, F, \perp_F)$ and $g_1, g_2 \notin F$ be first-order reducible for $\mathbb{K}_F$. Suppose $g_1 \perp g_2$.*

Both g_1, g_2 are irreducible in $\mathbb{K}_{F\cup G}$ $\quad\Leftrightarrow\quad$ ext$(C_1) \not\subseteq$ int(C_2).

Proof. As g_1 is irreducible it holds that $g_1^{\perp_{F\cup G}} \neq \text{ext}(C_1)^{\perp_{F\cup G}}$. From Corollary 3 follows that $g_1 \in \text{ext}(C_1)^{\perp_{F\cup G}}$ iff $g_1 \in g_1^{\perp_{F\cup G}}$. Therefore, $\text{ext}(C_1)^{\perp_{F\cup G}} = g_1^{\perp_{F\cup G}} \setminus \{g_2\}$. Hence, $\text{ext}(C_1) \not\perp g_2$, hence, $\text{ext}(C_1) \not\subseteq \text{int}(C_2)$. Similarly for g_2, $\text{ext}(C_2) \not\subseteq \text{int}(C_1)$.

Proposition 4. *Let $C_1, C_2 \in \mathfrak{B}(F, F, \perp_F)$ and $g_1, g_2 \notin F$ be first-order reducible for $\mathbb{K}_F$. Suppose $g_1 \not\perp g_2$.*

Both g_1, g_2 are irreducible in $\mathbb{K}_{F\cup G}$ $\quad\Leftrightarrow\quad$ ext$(C_1) \subseteq$ int(C_2).

Proof. As g_1 is irreducible it holds that $g_1^{\perp_{F\cup G}} \neq \text{ext}(C_1)^{\perp_{F\cup G}}$. From Corollary 3 follows that $g_1 \in \text{ext}(C_1)^{\perp_{F\cup G}}$ iff $g_1 \in g_1^{\perp_{F\cup G}}$. Therefore, $\text{ext}(C_1)^{\perp_{F\cup G}} = g_1^{\perp_{F\cup G}} \cup \{g_2\}$. Hence, $\text{ext}(C_1) \perp g_2$, hence, $\text{ext}(C_1) \subseteq \text{int}(C_2)$. By the properties of derivation operators, $\text{ext}(C_2) \subseteq \text{int}(C_1)$.

The functions mentioned in Propositions 4 and 3 can be called *second-order irreducible for $\mathbb{K}_F$*. In the next proposition we show that it is not necessary to look for three functions at once in order to find all p-indecomposable functions. Therefore, we do not need to define third-order irreducibility.

Here we use the notation: for $I \subseteq \{1, 2, 3\} : L_I = \{g_i \mid i \in I\}$. We omit the curly brackets in I, i.e. $L_{\{1,2\}} = L_{12} = \{g_1, g_2\}$.

Proposition 5. *Let $G = \{g_1, g_2, g_3\}$ be a set of functions such that $G \cap F = \emptyset$ and for $i \in \{1, 2, 3\} : g_i^{\perp_{F\cup G}} \cap F =$ int(C_i). If not all functions from G are reducible in $\mathbb{K}_{F\cup G}$ then there exists $L \subset G$ such that not all functions from L are reducible in $\mathbb{K}_{F\cup L}$.*

Proof. Let g_1 be reducible in $\mathbb{K}_{F\cup L_{12}}$ and in $\mathbb{K}_{F\cup L_{13}}$. Then there exists $H \subseteq F \cup \{g_2\} : H^{\perp_{F\cup L_{12}}} = g_1^{\perp_{F\cup L_{12}}}$ and $J \subseteq F \cup \{g_3\} : J^{\perp_{F\cup L_{13}}} = g_1^{\perp_{F\cup L_{13}}}$. Fix these H and J. If either g_2 is irreducible in $\mathbb{K}_{F\cup L_2}$ or g_3 is irreducible in $\mathbb{K}_{F\cup L_3}$ then the proposition is proved. Therefore, we can assume that they are reducible in corresponding context. Hence, without loss of generality, we can assume that $H, J \subseteq F$ (i.e. $H \cap G = J \cap G = \emptyset$). Note that

$$g_1^{\perp_{F\cup G}} = g_1^{\perp_{F\cup L_{13}}} \cup g_1^{\perp_{F\cup L_{12}}} = J^{\perp_{F\cup L_{13}}} \cup H^{\perp_{F\cup L_{12}}}. \tag{3}$$

Let $g_3 \in H^{\perp_{F\cup G}}$. Then $g_3 \perp H$. As $g_3^{\perp_{F\cup G}} \cap F = \text{int}(C_3)$ we obtain $H \subseteq \text{int}(C_3)$. Moreover, as $\text{int}(C_3)$ is an intent in $\mathbb{K}_F$ we have $H^{\perp_F\perp_F} \subseteq \text{int}(C_3)$. As $g_1^{\perp_{F\cup G}} \cap F = H^{\perp_F} = J^{\perp_F} = \text{int}(C_1)$ we have $J^{\perp_F\perp_F} \subseteq \text{int}(C_3)$ and, by

properties of closure operators, $J \subseteq \mathrm{int}(C_3)$. Therefore, $g_3 \perp J$ and $g_3 \in J^{\perp_{F\cup G}}$. Similarly, if $g_2 \in J^{\perp_{F\cup G}}$ then $g_2 \in H^{\perp_{F\cup G}}$. Hence,

$$H^{\perp_{F\cup L_{12}}} \cup J^{\perp_{F\cup L_{13}}} = H^{\perp_{F\cup G}} \cup J^{\perp_{F\cup G}}. \tag{4}$$

Combining (3) and (4) we obtain $g_1^{\perp_{F\cup G}} = H^{\perp_{F\cup G}} \cup J^{\perp_{F\cup G}}$. Therefore, $g_1^{\perp_{F\cup G}} = (H \cap J)^{\perp_{F\cup G}}$. Hence, g_1 is reducible in $\mathbb{K}_{F\cup G}$ and we arrive at a contradiction with initial assumption.

Therefore, if g_1, g_2 are in $\mathbb{K}_{F\cup L_{12}}$ then at least g_1 is irreducible in $\mathbb{K}_{F\cup L_{13}}$. If g_3 is reducible in $\mathbb{K}_{F\cup L_{13}}$ then g_1 is reducible in $\mathbb{K}_{F\cup L_1}$. Otherwise, both g_1, g_3 are irreducible in $\mathbb{K}_{F\cup L_{13}}$.

Suppose that a context $\mathbb{K}_F$ contains all p-indecomposable functions, however, the task is to prove this fact, i.e. that no further p-indecomposable functions exist. Suppose it has been checked that no counter-examples exist and every single function $g \in U_k$ is first-order reducible for $\mathbb{K}_F$. According to the above propositions it is necessary to look for exactly two functions at once in order to prove the desired statement. Therefore, in order to complete the proof for every $C_1, C_2 \in \mathfrak{B}(\mathbb{K}_F)$ one has to find all the functions g_1, g_2 such that $g_1^{\perp_{F\cup g_1}} \cap F = \mathrm{int}(C_1)$ and $g_2^{\perp_{F\cup g_2}} \cap F = \mathrm{int}(C_2)$ and then check if g_1 commutes with g_2. Therefore, one has to check the commutation property between all functions (if the context indeed contains all p-indecomposable functions). As already discussed, this task is infeasible. This result is discouraging. However, having the knowledge about the final result in some cases we can guarantee that all p-indecomposable functions will be found even without looking for two functions at once.

4.2 Implicatively Closed Subcontexts

During the exploration of p-clones one can discover such a subcontext of functions that no further function is a counter-example to existing implications. We shall say that such a subcontext is *implicatively closed*, meaning that all the valid in this subcontext implications are valid in the final context as well. Analysis of similar constructions can be found in [Gan07].

In order to guarantee the discovery of all p-indecomposable functions (success of exploration) it would suffice to find such a subcontext that it is neither implicatively closed nor contained in any other implicatively closed subcontext. Suppose the context $\mathbb{K}_F = (F, F, \perp_F)$, $F \subseteq U_k$ is discovered. As earlier, we denote the context of all p-indecomposable functions on U_k by $\mathbb{K}_{F_k^p}$. Let $S = F_k^p \backslash F$. It would be desirable to be able to guarantee the discovery of functions S by considering only the discovered part of relation $\perp_F$ and the part $\perp_{FS}$ ($=\perp_{SF}^{-1}$), see Figure 3. Unfortunately, as the next example shows, in general it is not possible.

Example 2. Consider the context in Figure 4. The context contains all the p-indecomposable functions from U_2 and three additional objects g_1, g_2, g_3. Functions with commutation properties as of g_1, g_2, g_3 do not exist. However, if functions with commutation properties as of g_1, g_2, g_3 existed then the functions g_1, g_2

	F	S
F	$\perp_F$	$\perp_{FS}$
S	$\perp_{SF}$	$\perp_S$

Fig. 3. Partitioning of the context $\mathbb{K}_{F_k^p}$ of all p-indecomposable functions

would not be counter-examples to any valid in $\mathbb{K}_{F_2^p \cup g_3}$ implication. Note that g_3 is a counter-example to a valid in $\mathbb{K}_{F_2^p}$ implication. Therefore, the subcontext containing functions $F_2^p \cup g_3$ would be implicatively closed. Moreover, it is even closed with respect to finding first-order irreducible functions as g_1 is reducible in $\mathbb{K}_{F_2^p \cup \{g_1, g_3\}}$ and g_2 is reducible in $\mathbb{K}_{F_2^p \cup \{g_2, g_3\}}$.

However, if instead of g_3 we consider the function g_4, which differs from g_3 only in that g_4 commutes with both g_1 and g_2, then the subcontext containing $F_2^p \cup g_4$ is neither implicatively closed nor contained in any implicatively closed subcontext of the context $\mathbb{K}_{F_2^p \cup \{g_1, g_2, g_4\}}$. The difference between g_3 and g_4 is contained in $\perp_S$ in Figure 3. Therefore, in general it is not possible to guarantee the discovery of functions S without considering $\perp_S$.

	f_0^u	f_1^u	f_{14}^b	f_8^b	f_{212}^t	f_{150}^t	f_3^u	g_3	g_4	g_1	g_2
f_0^u	×		×	×	×	×		×	×	×	
f_1^u		×			×	×					
f_{14}^b	×		×				×	×	×		
f_8^b	×			×			×				×
f_{212}^t	×	×					×	×	×		
f_{150}^t	×	×				×	×				
f_3^u			×	×	×	×	×	×	×	×	×
g_3	×		×		×		×				
g_4	×		×		×		×			**×**	**×**
g_1	×						×		**×**		×
g_2				×			×		**×**	×	×

Fig. 4. Context $\mathbb{K}_{F_2^p \cup \{g_1, g_2, g_3\}}$ from Example 2

Definition 2. *Let $\mathbb{K}_H$ be a context, $\mathbb{K}_F \subseteq \mathbb{K}_H$, $S = H \setminus F$. An object $s \in S$ is called an* essential counter-example *for $\mathbb{K}_F$ if there exists a valid in $\mathbb{K}_F$ implication Imp such that*

1. *s is a counter-example to Imp;*
2. *there does not exist an object* $p \in S \setminus \{s\}$ *such that p is a counter-example to Imp.*

It is clear that all the essential counter-examples will necessarily be added to the context during the exploration. The next proposition suggests how one can check if a counter-example is essential or not.

In the context $\mathbb{K}_{F_3^p}$ there are several pairs of functions (f_1, f_2) such that they commute with the same functions except for one commutes with itself and the other does not commute with itself. These functions cannot be essential counter-examples, because they are counter-examples to the same implications, if any. However, if they are the only counter-examples to some valid implication then these functions will eventually be discovered by object-attribute exploration.

Proposition 6. *Let* $s_1, s_2 \in S$ *such that* $s_2 \not\perp s_2$ *and* $s_1^{\perp_{U_k}} = s_2^{\perp_{U_k}} \cup \{s_2\}$. *If there exists a valid in* $\mathbb{K}_F$ *implication Imp such that the counter-examples are exactly* $s_1, s_2 \in S$ *then* s_1 *is first-order irreducible for* $\mathbb{K}_{F \cup s_2}$ *and* s_2 *is first-order irreducible for* $\mathbb{K}_{F \cup s_1}$.

Proof. s_1 **in** $\mathbb{K}_{F \cup s_2}$**.** As *Imp* is valid in $\mathbb{K}_F$ the set $s_2^{\perp_{F \cup s_1}}$ is closed in $\mathbb{K}_F$. Therefore, as follows from Proposition 1 for the object concept of s_2 ($\mathrm{ext}(C_{s_2}) \not\subseteq \mathrm{int}(C_{s_2})$), the function s_1 ($s_1 \perp s_1$) is first-order irreducible.

s_2 **in** $\mathbb{K}_{F \cup s_1}$**.** As *Imp* is valid in $\mathbb{K}_F$ the set $s_1^{\perp_{F \cup s_2}}$ is closed in $\mathbb{K}_F$. Therefore, as follows from Proposition 2 for the object concept of s_1 ($\mathrm{ext}(C_{s_1}) \subseteq \mathrm{int}(C_{s_1})$), the function s_2 ($s_2 \not\perp s_2$) is first-order irreducible.

We have investigated different types of reducibilities, we have shown, that there do not exist third-order irreducible functions. However, the task of finding second-order irreducible functions is infeasible. Fortunately, it is possible to find not only zero-order irreducible functions, but also first-order irreducible functions. Moreover, if it would be possible to prove that the functions undiscovered at the moment are not second-order irreducible then we can guarantee that all the p-indecomposable functions will eventually be discovered.

5 Results

We take all unary functions as the starting point. Thanks to earlier investigation in [Dan77] we know the final context. When we investigate all possible implicatively closed partitions such that the implicatively closed subcontext contains all unary functions we find the following:

- We start with 27 unary functions, 26 of them are p-indecomposable;
- After adding all essential counter-examples we obtain 147 functions;
- After using Proposition 6 we obtain 155 functions;
- There remain 42 functions to be discovered. By direct check we find that there does not exist an implicatively closed subcontext containing 155 mentioned above functions such that all the undiscovered functions are second-order irreducible.

Hence, if we start from all unary functions on A_3 all the functions F_3^p will eventually be discovered.

The experiment was conducted three times starting from different initial contexts, all three times the exploration was successful. The exploration stating from a single constant function $f_{3,0}^u$ took 207 steps.

References

[BW87] S. Burris and R. Willard. Finitely many primitive positive clones. *Proceedings of the American Mathematical Society*, 101(3):427–430, 1987.

[Dan77] A.F. Danil'chenko. Parametric expressibility of functions of three-valued logic. *Algebra and Logic*, 16(4):266–280, 1977.

[Dau00] F. Dau. Implications of properties concerning complementation in finite lattices. *In: Contributions to General Algebra 12 (D. Dorninger et al., eds.), Proceedings of the 58th workshop on general algebra "58. Arbeitstagung Allgemeine Algebra", Vienna, Austria, June 3-6, 1999, Verlag Johannes Heyn, Klagenfurt*, pages 145–154, 2000.

[Gan07] B. Ganter. Relational galois connections. *Formal Concept Analysis*, pages 1–17, 2007.

[GD86] J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Math. Sci. Hum*, 24(95):5–18, 1986.

[GW99] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, 1999.

[KPR06] L. Kwuida, C. Pech, and H. Reppe. Generalizations of boolean algebras. an attribute exploration. *Mathematica Slovaca*, 56(2):145–165, 2006.

[Kuz79] A.V. Kuznetsov. Means for detection of nondeducibility and inexpressibility. *Logical Inference*, pages 5–33, 1979.

[Pos42] E.L. Post. *The two-valued iterative systems of mathematical logic.* Princeton University Press, 1942.

[Rev14] A. Revenko. Automatized construction of implicative theory of algebraic identities of size up to 5. In Cynthia Vera Glodeanu, Mehdi Kaytoue, and Christian Sacarea, editors, *Formal Concept Analysis*, volume 8478 of *Lecture Notes in Computer Science*, pages 188–202. Springer International Publishing, 2014.

[Rev15] A. Revenko. *Automatic Construction of Implicative Theories for Mathematical Domains.* PhD thesis, TU Dresden, 2015.

[Ros70] I. Rosenberg. *Über die funktionale Vollständigkeit in den mehrwertigen Logiken: Struktur der Funktionen von mehreren Veränderlichen auf endlichen Mengen.* Academia, 1970.

[Yab60] S.V. Yablonsky. *Functional Constructions in K-valued Logic.* U.S. Joint Publications Research Service, 1960.

[YM59] Yu.I. Yanov and A.A. Muchnik. On the existence of k-valued closed classes that have no bases. *Doklady Akademii Nauk SSSR*, 127:44–46, 1959.

Putting OAC-triclustering on MapReduce

Sergey Zudin, Dmitry V. Gnatyshak, and Dmitry I. Ignatov

National Research University Higher School of Economics, Russian Federation
dignatov@hse.ru
http://www.hse.ru

Abstract. In our previous work an efficient one-pass online algorithm for triclustering of binary data (triadic formal contexts) was proposed. This algorithm is a modified version of the basic algorithm for OAC-triclustering approach; it has linear time and memory complexities. In this paper we parallelise it via map-reduce framework in order to make it suitable for big datasets. The results of computer experiments show the efficiency of the proposed algorithm; for example, it outperforms the online counterpart on Bibsonomy dataset with $\approx 800,000$ triples.

Keywords: Formal Concept Analysis, triclustering, triadic data, data mining, big data, MapReduce

1 Introduction

Mining of multimodal patterns is one of the hot topics in Data Mining and Machine Learning [1,2,3,4]. Thus, cluster analysis of multimodal data and specifically of dyadic and triadic relations is a natural extension of the idea of original clustering. In dyadic case biclustering methods (the term bicluster was coined in [5]) are used to simultaneously find subsets of the sets of objects and attributes that form homogeneous patterns of the input object-attribute data. In fact, one of the most popular applications of biclustering is gene expression analysis in Bionformatics [6,7]. Triclustering methods operate in triadic case where for each object-attribute pair one assigns a set of some conditions [8,9,10]. Both biclustering and triclustering algorithms are widely used in such areas as gene expression analysis [11,12,13], recommender systems [14,15,16], social networks analysis [17], etc. The processing of numeric multimodal data is also possible by modifications of existing approaches for mining dyadic binary relations [18].

Though there are methods that can enumerate all triclusters satisfying certain constraints [2] (in most cases they ensure that triclusters are dense), their time complexity is rather high, as in the worst case the maximal number of triclusters usually is exponential (e.g. in case of formal triconcepts), showing that these methods are hardly scalable. To process big data algorithms need to have at most linear time complexity (e.g., $O(|I|)$ in case of n-ary relation I) and be easily parallelisable. In addition, in most cases, it is necessary that such algorithms output the results in one pass.

Earlier, in order to create an algorithm satisfying these requirements, we adapted a triclustering method based on prime operators (prime OAC-triclustering

method) [10] and proposed its online version, which is linear, one-pass and easily parallelisable [19]. However, its parallelisation is possible in different ways. For example, one can use a popular framework for commodity hardware, MapReduce (M/R) [20]. By the way, there were several successful M/R implementations in the FCA community and other lattice-oriented domains. Thus, in [21], the authors adapted Close-by-One algorithm to M/R framework and showed its efficiency. At the same year, in [22], an efficient M/R algorithm for computation of closed cube lattices was proposed. The authors of [23] demonstrated that iterative algorithms like Ganter's NextClosure can benefit from the usage of iterative M/R schemes.

Note that experts aware potential users that M/R is like a big cannon that requires long preparations to shot but fires fast: "the entire distributed-file-system milieu makes sense only when files are very large and are rarely updated in place " [20]. In this work, in contrast to our previous study, we assume that there is a large bulk of data to process that are not coming online.

The rest of the paper is organized as follows: in Section 2, we recall the original method and the online version of the algorithm of prime OAC-triclustering. In Section 3, we describe the M/R setting of the problem and the corresponding M/R version of the original algorithm with important implementation aspects. Finally, in Section 4 we show the results of several experiments which demonstrate the efficiency of the M/R version of the algorithm. As an addendum, in the Appendix section, the reader may find our proposal for alternative models of M/R-based variants of prime OAC-triclustering.

2 Prime object-attribute-condition triclustering

Prime object-attribute-condition triclustering method (OAC-prime) based on Formal Concept Analysis [24,25] is an extension for the triadic case of object-attribute biclustering method [26]. Triclusters generated by this method have similar structure as the corresponding biclusters, namely the cross-like structure of triples inside the input data cuboid (i.e. formal tricontext).

Let $\mathbb{K} = (G, M, B, I)$ be a triadic context, where G, M, B are respectively the sets of objects, attributes, and conditions, and $I \subseteq G \times M \times B$ is a triadic incidence relation. Each prime OAC-tricluster is generated by applying the following prime operators to each pair of components of some triple:

$$\begin{aligned} (X,Y)' &= \{b \in B \,|\, (g,m,b) \in I \text{ for all } g \in X, m \in Y\}, \\ (X,Z)' &= \{m \in M \,|\, (g,m,b) \in I \text{ for all } g \in X, b \in Z\}, \\ (Y,Z)' &= \{g \in G \,|\, (g,m,b) \in I \text{ for all } m \in Y, b \in Z\}, \end{aligned} \tag{1}$$

where $X \subseteq G$, $Y \subseteq M$, and $Z \subseteq B$.

Then the triple $T = ((m,b)', (g,b)', (g,m)')$ is called *prime OAC-tricluster* based on triple $(g,m,b) \in I$. The components of tricluster are called, respectively, *tricluster extent*, *tricluster intent*, and *tricluster modus*. The triple (g,m,b) is called a *generating triple* of the tricluster T. Figure 1 shows the structure of an OAC-tricluster (X,Y,Z) based on triple $(\widetilde{g}, \widetilde{m}, \widetilde{b})$, triples corresponding to the

gray cells are contained in the context, other triples may be contained in the tricluster (cuboid) as well.

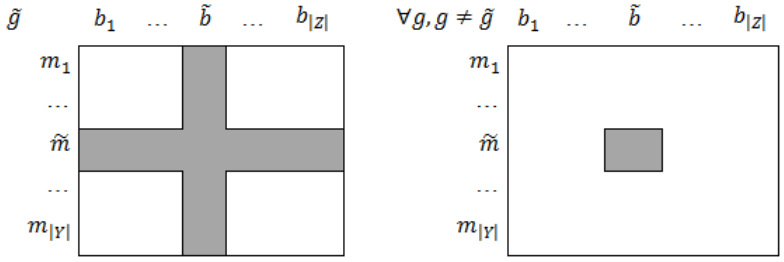


Fig. 1. Structure of prime OAC-triclusters: the dense cross-like central layer containing $\tilde{g}$ (left) and the layer for an object g (right) in $M \times B$ dimensions.

The basic algorithm for prime OAC-triclustering method is rather straightforward (see [10]). First of all, for each combination of elements from each two sets of $\mathbb{K}$ we apply the corresponding prime operator (we call the resulting sets *prime sets*). After that we enumerate all triples from I and on each step we must generate a tricluster based on the corresponding triple, check whether this tricluster is already contained in the tricluster set (by using hashing) and also check extra conditions.

The total time complexity of the algorithm depends on whether there is a non-zero minimal density threshold or not and on the complexity of the hashing algorithm used. In case we use some basic hashing algorithm processing the tricluster's extent, intent and modus without a minimal density threshold, the total time complexity is $O(|G||M||B| + |I|(|G| + |M| + |B|))$; in case of a non-zero minimal density threshold, it is $O(|I||G||M||B|)$. The memory complexity is $O(|I|(|G| + |M| + |B|))$, as we need to keep the dictionaries with the prime sets in memory.

In online setting, for triples coming from triadic context $\mathbb{K} = (G, M, B, I)$, the user has no a priori knowledge of the elements and even cardinalities of G, M, B, and I. At each iteration we receive some set of triples from I: $J \subseteq I$. After that we must process J and get the current version of the set of all triclusters. It is important in this setting to consider every pair of triclusters as being different as they have different generating triples, even if their respective extents, intents, and modi are equal. Thus, any other triple can change only one of these two triclusters, making them different.

To efficiently access prime sets for their processing, the dictionaries containing the prime sets are implemented as hash-tables.

The algorithm is straightforward as well (Alg. 1). It takes some set of triples (J), the current tricluster set ($\mathcal{T}$), and the dictionaries containing prime sets ($Primes$) as input and outputs the modified versions of the tricluster set and

dictionaries. The algorithm processes each triple (g, m, b) of J sequentially (line 1). At each iteration the algorithm modifies the corresponding prime sets (lines 2-4).

Finally, it adds a new tricluster to the tricluster set. Note that this tricluster contains pointers to the corresponding prime sets (in the corresponding dictionaries) instead of the copies of the prime sets (line 5) which allows to lower the memory and access costs.

Algorithm 1 Add function for the online algorithm for prime OAC-triclustering.

Input: J is a set of triples;
$\mathcal{T} = \{T = (*X, *Y, *Z)\}$ is a current set of triclusters;
$PrimesOA$, $PrimesOC$, $PrimesAC$.
Output: $\mathcal{T} = \{T = (*X, *Y, *Z)\}$;
$PrimesOA$, $PrimesOC$, $PrimesAC$.
1: **for all** $(g, m, b) \in J$ **do**
2: $PrimesOA[g, m] := PrimesOA[g, m] \cup \{b\}$
3: $PrimesOC[g, b] := PrimesOC[g, b] \cup \{m\}$
4: $PrimesAC[m, b] := PrimesAC[m, b] \cup \{g\}$
5: $\mathcal{T} := \mathcal{T} \cup \{(\&PrimesAC[m, b], \&PrimesOC[g, b], \&PrimesOA[g, m])\}$
6: **end for**

The algorithm is one-pass and its time and memory complexities are $O(|I|)$.

Duplicate elimination and selection patterns by user-specific constraints are done as post-processing to avoid patterns' loss. The time complexity of the basic post-processing is $O(|I|)$ and it does not require any additional memory.

Finally, it seems the algorithm can be easily parallelised by splitting the subset of triples J into several subsets, processing each of them independently, and merging the resulting sets afterward.

3 Map-reduce OAC-triclustering

3.1 Map-reduce decomposition

We use a two-stage M/R approach. The first M/R allows us to efficiently calculate all the primes of the existed pairs. The second M/R permits to assemble the found primes into triclusters. During the first map phase, each triple from the input context is indexed by a key using hash function depending on one of the basic entity types, object, attribute, or condition (see Alg. 2). The number of map keys is equal to the number of reducers.

Then each first reducer receives the portion of data for a particular key (see Alg. 3). The internal reducer algorithm is almost a replication of Online OAC-prime. However, it does not assemble all found triclusters into a final collection; the reducer simply writes the file with the current triclusters for a given portion of data to a file or pass it to the second-stage mapper. Since in Hadoop MapReduce

Algorithm 2 Distributed OAC-triclustering: First Map

Input: S is a set of input triples as strings;
r is a number of reducers;
i is a grouping index (objects, attributes or conditions).
Output: $\tilde{J}$ is a list of $\langle key, triple \rangle$ pairs.

```
1: for all s ∈ S do
2:     t := transform(s)
3:     key := hash(t[i]) mod r
4:     J̃ := J̃ ∪ {⟨key, t⟩}
5: end for
```

we should work with text input files and our data are mainly in a tuple-based form, we use encode/decode function $encode()/transform()$ to switch between the internal tuple representation and the text-based one.

Algorithm 3 Distributed OAC-triclustering: First Reduce

Input: J is a list of triples (for a certain key);
$\mathcal{T} = \{T = (X, Y, Z)\}$ is a current set of triclusters;
$PrimesOA$, $PrimesOC$, $PrimesAC$.
Output: file of strings – encoded $\langle triple, tricluster \rangle$ pairs.

```
1: Primes ← initialise a new multimap
2: for all (g, m, b) ∈ J do
3:     Primes[g, m] := Primes[g, m] ∪ {b}
4:     Primes[g, b] := Primes[g, b] ∪ {m}
5:     Primes[m, b] := Primes[m, b] ∪ {g}
6: end for
7: for all (g, m, b) ∈ J do
8:     T := (set(Primes[m, b]), set(Primes[g, b]), set(Primes[g, m]))
9:     s := {encode(⟨(g, m, b), T⟩)}
10:    store s
11: end for
```

The second mapper takes the found intermediate triclusters (with their keys) as strings from the files produced by the first-stage reducers (see Alg. 4). It fills $Primes$ multimap in one pass through all $\langle triple, tricluster \rangle$ pairs. In the next loop for each key (g, m, b) the corresponding tricluster is formed and $\langle tricluster, tricluster \rangle$ pairs are passed to the second-stage reducer (the key $tricluster$ can be efficiently implemented by a proper hashing). In its turn, the second stage reducer eliminates duplicates and outputs the resulting file (Alg. 4). The $set()$ function helps to avoid duplicates among the values of $Primes[,]$, which is closer to our implementation. However, one can easily omit $set()$ in line 8, provided that $Primes$ is properly implemented.

The time complexity of the M/R solution is composed from two terms for each stage: $O(|I|/r)$ and $O(|I|)$. However, there are communication costs that

Algorithm 4 Distributed OAC-triclustering: Second Map

Input: S is a list of strings.
Output: $\tilde{\mathcal{T}}$ is an list of $\langle tricluster, tricluster\rangle$ pairs.

```
Primes ← initialise a new multimap
for all s ∈ S do
    ⟨(g, m, b), T⟩ := decode(s)
    update Primes multimap appropriately
    I := I ∪ {(g, m, b)}
end for
for all (g, m, b) ∈ I do
    T := (set(Primes[m, b]), set(Primes[g, b]), set(Primes[g, m]))
    T̃ := T̃ ∪ {⟨T, T⟩}
end for
```

Algorithm 5 Distributed OAC-triclustering: Second Reduce

Input: $\hat{\mathcal{T}}$ is a list of $\langle tricluster, list\ of\ triclusters\rangle$ pairs.
Output: File with a final set of triclusters $\{T = (X, Y, Z)\}$.

```
for all ⟨T, [T, ..., T]⟩ ∈ T̂ do
    store T
end for
```

should be inevitably paid and can be theoretically estimated as follows [20]: the replication rate for the first M/R stage $r_1 = 1$ (each triple is passed as one key-value pair), the reducer size $q_1 = |I|/r$; the replication rate for the second M/R stage is $r_2 = 1$ (it assign one key-value pair for each tricluster), but the reducer size varies from $q_2^{min} = 1$ (no duplicate triclusters) and $q_2^{max} = |I|$ (one final tricluster when all the initial triples belong to one absolutely dense cuboid).

3.2 Implementation aspects and used technologies

The application [1] has been implemented in Java within JRE 8 and as distributed computation framework we use Apache Hadoop [2].

We have used many other technologies: Apache Maven (framework for automatic project assembling), Apache Commons (for work with extended Java collections), Google Guava (utilities and data structures), Jackson JSON (open-source library for transformation of object-oriented representation of an object like tricluster to string), TypeTools (for real-time type resolution of inbound and outbound key-value pairs), etc.

ChainingJob module. During the development we found that in Hadoop one MapReduce process can contain only one Mapper and one Reducer. Thus, in order to develop an application with three "map" phases and one "reduce", one needs to create three processes. One process creation (even without various adjustments) takes 8-10 lines of code. After our vain search of an appropriate

[1] `https://github.com/zydins/DistributedTriclustering`
[2] `http://hadoop.apache.org/`

library, we developed "chaining-job" module [3]. Its main class contains the following fields: "jobs" (list of all scheduled processes), "name" (common name for all processes), and "tempDir" (folder name for intermediate results). First, the algorithm set input path for the first chaining process and path to the result of the last job; the rest jobs are connected by input and output "key-value" pairs and directory for intermediate files storage. Then this algorithm runs processes according to the schedule and waits their completion. In other words, it connects the input and output of chaining processes that run sequentially.

Let us shortly describe the most important classes our M/R implementation.

Entity. It is a basic class for object oriented representation of input strings and maintains three entity types: EXTENT, INTENT, and MODUS. For example: {"Leon", EXTENT }.

Tuple. An object of this class stores references to objects of class Entity and represents two basic entities: triple and tricluster. Mapper and Reducer classes operate with objects of this type.

FormalContext. This class is an object oriented representation of the underlying binary relation; it keeps the reference to an object of EntityStorage class (see below). It also contains methods "add" (add triple) and "getTriclusters" (get the output set of unique triclusters).

EntityStorage. This class manages the work with extents, intents and modi of triclustes. It also contains three dictionaries with composite keys. For example, for $(g1, m1, c1)$ object $c1$ will be added by key $(g1, m1)$ to the first dictionary; analogously for keys $(g1, c1)$ and $(m1, c1)$.

The process-like M/R classes are summarised below.

TupleReadMapper. Its main goal is reading a triple from the input file and transform the triple to an object of class Tuple.

TupleContextReducer. It receives input tuples and fills the underlying tricontext by them. It also sets the number of first reducers. This number depends on the available nodes in a distributed system and the structure of input data. The more unique entities are in triples, the more that value should be.

PrepareMapper. The "map" method receives files from the previous stage. They contain intermediate triclusters from each object of class TupleContextReducer. It fills the dictionary with primes. Further, each tricluster triple is transformed to Tuple structure and is passed to the second reduce phase.

CollectReduce. This class gathers all intermediate triclusters and obtains the final tricluster set. This process runs in several threads for speed up. The number of threads is a user-specified parameter.

Executor. It is a starting class of the application, which receives the input parameters, activates "chaining-job" utility for making a chain of jobs, and starts the execution.

[3] `https://github.com/zydins/chaining-job`

4 Experiments

Two series of experiments have been conducted in order to test the application on the synthetic contexts and real world datasets with moderate and large number of triples in each. In each experiment both versions of the OAC-triclustering algorithm have been used to extract triclusters from a given context. Only online and M/R versions of OAC-triclustering algorithm have managed to result patterns for large contexts since the computation time of the compared algorithms was too high (>3000 s). To evaluate the runtime more carefully, for each context the average result of 5 runs of the algorithms has been recorded.

4.1 Datasets

Synthetic datasets. As it was mentioned, synthetic contexts were randomly generated: 1) 20,000 triples (25 unique entities of each type); 2) 100,000 triples (50 unique entities of each type); 3) 1,000,000 triples (all possible combinations of 100 unique entities of each type). However, it is easy to see that some datasets are not correct formal contexts from algebraic viewpoint. Thus, the first dataset inevitably contains duplicates since $25 \times 25 \times 25$ gives only 15,625 unique triples. The second one contains less triples than $50^3 = 125,000$, the number of all possible combinations. The third one is just an absolutely dense cuboid $100 \times 100 \times 100$ (it contains only one formal concept (OAC-tricluster), the whole context).

These tests look more like crush test, but they have sense since in M/R setting the triples can be (partially) repeated, e.g., because of M/R task failures on some nodes (i.e. restarting processing of some key-value pairs). Even though the third dataset does not result in $3^{min(|G|,|M|,|B|)}$ formal triconcepts, the worst case for formal triconcepts generation in terms of the number of patterns, this is an example of the worst case scenario for the second reducer since its size is maximal ($q_2^{max} = |I|$). By the way, our algorithm should correctly assemble the only one tricluster (G, M, B) and it actually does.

IMDB. This dataset consists of Top-250 list of the Internet Movie Database (250 best movies based on user reviews). The following triadic context is composed: the set of objects consists of movie names, the set of attributes (tags), the set of conditions (genres), and each triple of the ternary relation means that the given movie has the given genre and is assigned the given tag.

Bibsonomy. Finally, a sample of the data of bibsonomy.org from ECML PKDD discovery challenge 2008 has been used. This website allows users to share bookmarks and lists of literature and tag them. For the tests the following triadic context has been prepared: the set of objects consists of users, the set of attributes (tags), the set of conditions (bookmarks), and a triple of the ternary relation means that the given user has assigned the given tag to the given bookmark.

The table 1 contains the summary of the contexts.

Table 1. Contexts for the experiments

Context	$\|G\|$	$\|M\|$	$\|B\|$	# triples	Density
20k	25	25	25	20,000	1
100k	50	50	50	100,000	0.8
1m	100	100	100	1,000,000	1
IMDB	250	795	22	3,818	0.00087
BibSonomy	2,337	67,464	28,920	816,197	$1.8 \cdot 10^{-7}$

4.2 Results

The experiments has been conducted on the computer running under OS X 10, using 1,8 GHz Intel Core i5, having 4 Gb 1600 MHz DDR3 and having 8 Gb free space on its hard drive (a typical commodity hardware). Two M/R modes have been tested: sequential mode of tasks completion and emulation of distributed one with 16 first reducers and 32 threads for the second stage.

Table 2. Results of comparison (time is given in seconds)

Algorithm/Context	IMDB (≈3k triples)	20k triples	100k triples	1m triples	Bibsonomy (≈800k triples)
Tribox	324	800	1,265	>3,000	>3,000
TRIAS	189	362	862	>3,000	>3,000
OAC Box	374	756	1,265	>3,000	>3,000
OAC Prime	7	8	734	>3,000	>3,000
Online OAC prime	3	3	3	5	>3,000
M/R OAC prime seq.	12	30	81	166	1,534
M/R OAC prime distr.	1	15	20	25	520

In Table 2 we summarise the results of performed tests. It is clear that on average our application has fewer execution time than its competitors, except of online version of OAC-triclustering. If we compare the implemented program with its original online version, the results are worse for not that big but dense datasets (closer to the worst case scenario $q_2 = |I|$. It is the consequence of the fact that the application architecture aimed at processing of large amounts of data; in particular, it is implemented in two stages with time consuming communication. Launching and stopping Apache Hadoop, data writing and passing between Map and Reduce steps in both stages requires substantial time, that is why for not that big datasets, when execution time is comparable with time for infrastructure management, time performance is not perfect. However, with data size increase the relative performance is growing. Thus, the last test for BibSonomy data has been successfully passed, but the competitors were not able to finish it within 50 min, but our M/R program did it even in sequential mode within 25 min.

5 Conclusion

In this paper we have presented a map-reduce version of OAC-triclustering algorithm. We have shown that the algorithm is efficient from both theoretical and practical points of view. It remains of linear time complexity and is performed in two stages (with each stage being M/R distributed); this allows us to use it for big data problems. However, we believe that it is possible to propose another variants of map-reduce based algorithm where the reducer exploits composite keys directly (see Appendix section). So, such algorithms and their comparison with the current M/R version on real and artificial data is still be in our plans. However, in despite the step towards Big Data technologies, a proper comparison of the proposed OAC triclustering and noise tolerant patterns in n-ary relations by DataPeeler and its descendants [2] is not yet conducted.

Acknowledgments. The study was implemented in the framework of the Basic Research Program at the National Research University Higher School of Economics in 2014-2015, in the Laboratory of Intelligent Systems and Structural Analysis. The last two authors were partially supported by Russian Foundation for Basic Research, grant no. 13-07-00504. The authors would like to thank Yuri Kudriavtsev from PM-Square and Dominik Slezak from Infobright and Warsaw University for their encouragement given to our studies of M/R technologies.

References

1. Georgii, E., Tsuda, K., Schölkopf, B.: Multi-way set enumeration in weight tensors. Machine Learning **82**(2) (2011) 123–155
2. Cerf, L., Besson, J., Nguyen, K.N., Boulicaut, J.F.: Closed and noise-tolerant patterns in n-ary relations. Data Min. Knowl. Discov. **26**(3) (2013) 574–619
3. Spyropoulou, E., De Bie, T., Boley, M.: Interesting pattern mining in multi-relational data. Data Mining and Knowledge Discovery **28**(3) (2014) 808–849
4. Ignatov, D.I., Gnatyshak, D.V., Kuznetsov, S.O., Mirkin, B.: Triadic formal concept analysis and triclustering: searching for optimal patterns. Machine Learning (2015) 1–32
5. Mirkin, B.: Mathematical Classification and Clustering. Kluwer, Dordrecht (1996)
6. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: A survey. IEEE/ACM Trans. Comput. Biology Bioinform. **1**(1) (2004) 24–45
7. Eren, K., Deveci, M., Kucuktunc, O., Catalyurek, Umit V.: A comparative analysis of biclustering algorithms for gene expression data. Briefings in Bioinform. (2012)
8. Mirkin, B.G., Kramarenko, A.V.: Approximate bicluster and tricluster boxes in the analysis of binary data. In Kuznetsov, S.O., et al., eds.: RSFDGrC 2011. Volume 6743 of Lecture Notes in Computer Science., Springer (2011) 248–256
9. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J., Zhukov, L.E.: Can triconcepts become triclusters? International Journal of General Systems **42**(6) (2013) 572–593
10. Gnatyshak, D.V., Ignatov, D.I., Kuznetsov, S.O.: From triadic FCA to triclustering: Experimental comparison of some triclustering algorithms. In: CLA. (2013) 249–260

11. Zhao, L., Zaki, M.J.: Tricluster: An effective algorithm for mining coherent clusters in 3d microarray data. In: SIGMOD 2005 Conference. (2005) 694–705
12. Li, A., Tuck, D.: An effective tri-clustering algorithm combining expression data with gene regulation information. Gene regul. and syst. biol. **3** (2009) 49–64
13. Kaytoue, M., Kuznetsov, S.O., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in formal concept analysis. Inf. Sci. **181**(10) (2011) 1989–2001
14. Nanopoulos, A., Rafailidis, D., Symeonidis, P., Manolopoulos, Y.: Musicbox: Personalized music recommendation based on cubic analysis of social tags. IEEE Transactions on Audio, Speech & Language Processing **18**(2) (2010) 407–412
15. Jelassi, M.N., Yahia, S.B., Nguifo, E.M.: A personalized recommender system based on users' information in folksonomies. In Carr, L., et al., eds.: WWW (Companion Volume), ACM (2013) 1215–1224
16. Ignatov, D.I., Nenova, E., Konstantinova, N., Konstantinov, A.V.: Boolean Matrix Factorisation for Collaborative Filtering: An FCA-Based Approach. In: AIMSA 2014, Varna, Bulgaria, Proceedings. Volume LNCS 8722. (2014) 47–58
17. Gnatyshak, D.V., Ignatov, D.I., Semenov, A.V., Poelmans, J.: Gaining insight in social networks with biclustering and triclustering. In: BIR. Volume 128 of Lecture Notes in Business Information Processing., Springer (2012) 162–171
18. Kaytoue, M., Kuznetsov, S.O., Macko, J., Napoli, A.: Biclustering meets triadic concept analysis. Ann. Math. Artif. Intell. **70**(1-2) (2014) 55–79
19. Gnatyshak, D.V., Ignatov, D.I., Kuznetsov, S.O., Nourine, L.: A one-pass triclustering approach: Is there any room for big data? In: CLA 2014. (2014)
20. Rajaraman, A., Leskovec, J., Ullman, J.D.: MapReduce and the New Software Stack. In: Mining of Massive Datasets. Cambridge University Press, England, Cambridge (2013) 19–70
21. Krajca, P., Vychodil, V.: Distributed algorithm for computing formal concepts using map-reduce framework. In: N. Adams et al. (Eds.): IDA 2009. Volume LNCS 5772. (2009) 333–344
22. Kuznecov, S., Kudryavcev, Y.: Applying map-reduce paradigm for parallel closed cube computation. In: 1st Int. Conf. on Advances in Databases, Knowledge, and Data Applications, DBKDS 2009. (2009) 62–67
23. Xu, B., de Frein, R., Robson, E., Foghlu, M.O.: Distributed formal concept analysis algorithms based on an iterative mapreduce framework. In Domenach, F., Ignatov, D., Poelmans, J., eds.: ICFCA 2012. Volume LNAI 7278. (2012) 292–308
24. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. In Rival, I., ed.: Ordered Sets. Volume 83 of NATO Advanced Study Institutes Series. Springer Netherlands (1982) 445–470
25. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. 1st edn. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1999)
26. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J.: Concept-based biclustering for internet advertisement. In: ICDM Workshops, IEEE Computer Society (2012) 123–130

Appendix. Alternative variants of two-stage MapReduce

First Map: Finding primes. During this phase every input triple (g, m, b) is encoded by three key-value pairs $\langle (g, m), b\rangle$, $\langle (g, b), m\rangle$, and $\langle (m, b), g\rangle$. These pairs are passed to the first reducer. The replication rate is $r_1 = 3$.

First Reduce: Finding primes. This reducer fills three corresponding dictionaries for primes of keys. So, for example, the first dictionary, $PrimeOA$ contains key-value pairs $\langle (g,m), \{b_1, b_2, \ldots, b_n\}\rangle$. The reducer size is $q_1 = max(|G|, |M|, |B|)$

The process can be stopped after the first reduce phase and all the triclusters found as $(Prime[g,m], Prime[g,b], Prime[m,b])$ each by enumeration of $(g,m,b) \in I$. However, to do it faster and keep the result for further computation, it is possible to use M/R as well.

Second Map: Tricluster generation. The second map does tricluster combining job, i.e. for each triple (g,m,b) it composes the new key-value pair, $\langle (g,m,b), \emptyset\rangle$. And for each pair of either type, $\langle (g,m), Prime[g,m]\rangle$, $\langle (g,b), Prime[g,b]\rangle$, and $\langle (m,b), Prime[m,b]\rangle$ it generates key-values pairs $\langle (g,m,\tilde{b}), Prime[g,m]\rangle$, $\langle (g,\tilde{m},b), PrimeOC[g,b]\rangle$, and $\langle (\tilde{g},m,b), Prime[m,b]\rangle$, where $\tilde{g} \in G$, $\tilde{m} \in M$, and $\tilde{b} \in B$. $r_2 = (|I|+3|G||M||B|)/(|I|+|G||M|+|G||B|+|M||B|) \leq (\rho+3)/(\rho+3/max(|G|,|M|,|B|))$, where ρ is the input tricontext density.

Second Reduce: Tricluster generation. The second reducer just assembles only one value for each key (g,m,b), the generating triple, its tricluster, $(Prime[g,m], Prime[g,b], Prime[m,b])$. If there is no key-value pair $\langle (g,m,b), \emptyset\rangle$ for a particular triple (g,m,b), it does not output any key-value pair for the key. The reducer size q_2 is either 3 (no output) or 4 (tricluster assembled).

Second Map: Tricluster generation with duplicate generating triples. Second map does tricluster combining job, i.e. for each triple (g,m,b) it composes a new key-value pair: $\langle (Prime[g,m], Prime[g,b], Prime[m,b]), (g,m,b)\rangle$.

Second Map: Tricluster generation with duplicate generating triples. The second reducer just groups values for each key: $\langle (X,Y,Z), \{(g_1,m_1,b_1), \ldots, (g_n,m_n,b_n)\}\rangle$.

These two variations of the second stage have their merits: the first one is beneficial for further computations with a new portion of triples and the last one is more compact and informative. Of course, each variant of the second stage has its own runtime complexity which depends not only on the model representation but is also sensitive to datastructures implementation and M/R communication costs and settings.

Concept interestingness measures: a comparative study

Sergei O. Kuznetsov[1] and Tatiana P. Makhalova[1,2]

[1]National Research University Higher School of Economics, Kochnovsky pr. 3, Moscow 125319, Russia
[2]ISIMA, Complexe scientifique des Cézeaux, 63177 Aubière Cedex, France

skuznetsov@hse.ru,t.makhalova@gmail.com

Abstract. Concept lattices arising from noisy or high dimensional data have huge amount of formal concepts, which complicates the analysis of concepts and dependencies in data. In this paper, we consider several methods for pruning concept lattices and discuss results of their comparative study.

1 Introduction

Formal Concept Analysis (FCA) underlies several methods for rule mining, clustering and building taxonomies. When constructing a taxonomy one often deals with high dimensional or/and noisy data which results in a huge amount of formal concepts and dependencies given by implications and association rules. To tackle this issue different approaches were proposed for selecting most important or interesting concepts. In this paper we consider existing approaches which fall into the following groups: pre-processing of a formal context, modification of the closure operator, and concept filtering based on interestingness indices (measures). We mostly focus on comparison of interestingness measures and study their correlations.

2 FCA framework

Here we briefly recall FCA terminology [20]. A formal context is a triple (G, M, I), where G is called a set objects, M is called a set attributes and $I \subseteq G \times M$ is a relation called incidence relation, i.e. $(g, m) \in I$ if the object g has the attribute m. The derivation operators $(\cdot)'$ are defined for $A \subseteq G$ and $B \subseteq M$ as follows:

$$A' = \{m \in M | \forall g \in A : gIm\}$$
$$B' = \{g \in G | \forall m \in B : gIm\}$$

A' is the set of attributes common to all objects of A and B' is the set of objects sharing all attributes of B. The double application of $(\cdot)'$ is a closure operator,

i.e. $(\cdot)''$ is extensive, idempotent and monotone. Sets $A \subseteq G$, $B \subseteq M$, such that $A = A''$ and $B = B''$ are said to be closed.

A (formal) concept is a pair (A, B), where $A \subseteq G$, $B \subseteq M$ and $A' = B$, $B' = A$. A is called the (formal) extent and B is called the (formal) intent of the concept (A, B). A partial order $\leqslant$ is defined on the set of concepts as follows: $(A, B) \leq (C, D)$ iff $A \subseteq C\,(D \subseteq B)$, a pair (A, B) is a subconcept of (C, D), while (C, D) is a superconcept of (A, B).

3 Methods for simplifying a lattice structure

With the growth of the dimension of a context the size of a lattice can increase exponentially, it becomes almost impossible to deal with the huge amount of formal concepts. With this respect a wide variety of methods have been proposed. Classification of them was presented in [16]. Authors proposed to divide techniques for lattice pruning into three classes: redundant information removal, simplification, selection. In this paper, we consider also other classes of methods and their application to concept pruning.

3.1 Pre-processing

Algorithms for concept lattice are time consuming. To decrease computation costs one can reduce the size of a formal context. Cheung and Vogel [13] applied Singular Value Decomposition (SVD) to obtain a low-rank approximation of Term-Document matrix and construct concept lattice using pruned concepts. Since this method is also computationally complex [25], alternative methods such as spherical k-Means [14] and fuzzy k-Means [17], Non-negative Matrix Decomposition [33] were proposed.

Dimensionality reduction can dramatically decrease the computational load and simplify the lattice structure, but in most cases it is very difficult to interpret the obtained results.

Another way to solve described problems without changing the dimension of the context was proposed in [18], where an algorithm that significantly improves the lattice structure by making small changes of context was presented. The central notion of the method is the concept incomparability w.r.t. $\leq$ relation. The goal of the proposed method is to diminish total incomparability of the concepts in the lattice.

The authors note that the result is close to that of fuzzy k-Means, but the former is achieved with fewer context changes than required by the latter. However, such transformations do not always lead to the decrease of a number of formal concepts, the transformations of a context are aimed at increasing the share of comparable concepts, thus this method does not ensure a significant simplification of the lattice structure.

Context pruning by clustering objects was introduced in [15]. The similarity of objects is defined as the weighted sum of shared attributes. Thus, the original context is replaced by the reduced one. Firstly, we need to assign weights w^m

for each attribute $m \in M$. The similarity between objects is defined as weighted sum of shared attributes.

Objects are considered similar if $sim(g, h) \geq \varepsilon$, where ε is a predefined threshold. In order to avoid the generation of large clusters another threshold α was proposed. Thus, the algorithm is an agglomerative clustering procedure, such that at each step clusters are brought together if the similarity between them is less than ε and the volume of clusters is less than $\alpha|G|$ objects.

3.2 Reduction based on a background knowledge or predefined constraints

Another approach to tackle computation and representation issues is to determine constraints on the closure operator. It can be done using background knowledge of attributes. In [8] the extended closure operator was presented. It is based on the notion of AD-formulas (attribute-dependency formulas), which establish dependence of attributes and their relative importance. Put differently, the occurrence of certain attributes implies that more important ones should also occur. Concepts which do not satisfy this condition are not included in the lattice.

In [5] a numerical approach to defining attribute importance was proposed. The importance of a formal concept can be defined by various aggregation functions (average, minimum, maximum) and different intent subsets (generator, minimal generator or intent itself). It was shown [5] that there is a correspondence between this numerical approach and AD-formulas.

Carpineto and Romano [12] considered document-term relation and proposed to use a thesaurus of terms to prune the lattice. Two different attributes are considered as same if there is a common ancestor in the hierarchy. To enrich the set of attributes they used a thesaurus, but in general, it may be quite difficult to establish such kind of relationship between arbitrary attributes.

Computing concepts with extents exceeding a threshold was proposed in [26] and studied in relation to frequent itemset mining in [34]. The main drawback of this approach, called "iceberg lattice" mining, is missing rare and probably interesting concepts.

Several polynomial-time algorithms for computing Galois sub-hierarchies were proposed, see [9, 3].

3.3 Filtering concepts

Selecting most interesting concepts by means of interestingness measures (indices) is the most widespread way of dealing with the huge number of concepts. The situation is aggravated by complexity of computing some indices. However, this approach may be fruitful, since it provides flexible tools for exploration of a derived taxonomy. In this section we consider different indices for filtering formal concepts. These indices can be divided into the following groups: measures designed to assess closed itemsets (formal concepts), arbitrary itemsets and

measures for assessing the membership in a basic level (a psychology-motivated approach).

Indices for formal concepts

Stability Stability indices were introduced in [27, 28] and modified in [29]. One distinguishes intensional and extensional stability. The first one allows estimating the strength of dependence of an intent on each object of the respective extent. Extensional stability is defined dually.

$$Stab_i\left(A,B\right)=\frac{|\left\{C\subseteq A|C'=B\right\}|}{2^{|A|}}$$

The problem of computing stability is $\#P$-complete [28] and hence it makes this measure impractical for large contexts. In [4] its Monte Carlo approximation was introduced, a combination of Monte Carlo and upper bound estimate was proposed in [10]. Since for large contexts the stability is close to 1 [21] the logarithmic scale of stability (inducing the same ranking as stability) [10] is often used:

$$LStab\left(c\right)=-log_2\left(1-Stab\left(c\right)\right)$$

The bounds of stability are given by

$$\Delta_{min}\left(c\right)-log_2\left(|M|\right)\leq-log_2\sum_{d\in DD(c)}2^{-\Delta(c,d)}\leq LStab\left(c\right)\leq\Delta_{min}\left(c\right),$$

where $\Delta_{min}\left(c\right)=min_{d\in DD(c)}\Delta\left(c,d\right)$, $DD\left(c\right)$ is a set of all direct descendants of c in the lattice and $\Delta\left(c,d\right)$ is the size of the set-difference between extents of formal concepts c and d.

In our experiments we used the bounds of logarithmic stability, because the combined method is still computationally demanding.

Concept Probability Stability of a formal concept may be interpreted as probability of retaining its intent after removing some objects from the extent, taking that all subsets of the extent have equal probability. In [24] it was noticed that some interesting concepts with small number of object usually have low stability value. To ensure selection of interesting infrequent closed patterns, the concept probability was introduced. It is equivalent to the probability of a concept introduced earlier by R. Emilion [19].

The probability that an arbitrary object has all attributes from the set B is defined as follows

$$p_B=\prod_{m\in B}p_m$$

Concept probability is defined as the probability of B being closed:

$$p\left(B=B''\right)=\sum_{k=0}^{n}p\left(|B'|=k,B=B''\right)=\sum_{k=0}^{n}\left[p_B^k\left(1-p_B\right)^{n-k}\prod_{m\notin B}\left(1-p_m^k\right)\right]$$

where $n = |G|$.

The concept probability has the following probabilistic components: the occurrence of each attribute from B in all k objects, the absence of at least one attribute from B in other objects and the absence of other attributes shared by all k objects.

Robustness Another probabilistic approach to assessing a formal concept was proposed in [35]. Robustness is defined as the probability of a formal concept intent remaining closed while deleting objects, where every object of a formal context is retained with probability α. Then for a formal concept $c = (A, B)$ the robustness is given as follows:

$$r(c, \alpha) = \sum_{d \preceq c} (-1)^{|B_d| - |B_c|} (1 - \alpha)^{|A_c| - |A_d|}$$

Separation The separation index was considered in [24]. The main idea behind this measure is to describe the area covered by a formal concept among all nonzero elements in the corresponding rows and columns of the formal context. Thus, the value characterizes how specific is the relationship between objects and attributes of the concept with respect to the formal context.

$$\mathfrak{s}(A, B) = \frac{|A||B|}{\sum_{g \in A} |g'| + \sum_{m \in B} |m'| - |A||B|}$$

Basic Level Metrics The group of so-called "basic level" measures was considered by Belohlavek and Trnecka [6, 7]. These measures were proposed to formalize the existing psychological approach to defining basic level of a concept [31].

Similarity approach (S) A similarity approach to basic level was proposed in [32] and subsequently formalized and applied to FCA in [6]. The authors defined basic level as combination of three fuzzy functions that correspond to formalized properties outlined by Rosch: high cohesion of concepts, considerably greater cohesion with respect to upper neighbor and a slightly less cohesion with respect to lower neighbors. The membership degree of a basic level is defined as follows:

$$BL_S = coh^{**}(A, B) \otimes coh^{**}_{un}(A, B) \otimes coh^{**}_{ln}(A, B),$$

where α_i is a fuzzy function that corresponds to the conditions defined above, $\otimes$ is t-norm [23].

A cohesion of a formal concept is a measure of pairwise similarity of all object in the extent. Various similarity measures can be used for cohesion functions:

$$sim_{SMC}(B_1, B_2) = \frac{|B_1 \cap B_2| + |M - (B_1 \cup B_2)|}{|M|}$$

$$sim_J(B_1, B_2) = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|}$$

The first similarity index sim_{SMC} takes into account the number of common attributes, while Jaccard similarity sim_J takes exactly the proportion of attributes shared by two sets. There are two ways to compute cohesion of formal concepts: taking average or minimal similarity among sets of attributes of the concept extent, the formulas are represented below (for average and minimal similarity respectively).

$$coh^{a}_{...}(A,B) = \frac{\sum_{x_1,x_2 \subseteq A, x_1 \neq x_2} sim_{...}(x'_1, x'_2)}{|A|(|A|-1)/2}$$

$$coh^{m}_{...}(A,B) = \min_{x_1,x_2 \in A} sim_{...}(x'_1, x'_2)$$

The Rosch's properties for upper and lower neighbors take the following forms:

$$coh^{a*}_{...,un}(A,B) = 1 - \frac{\sum_{c \in UN(A,B)} coh^{*}_{...}(c)/coh^{*}_{...}(A,B)}{|UN(A,B)|}$$

$$coh^{a*}_{...,ln}(A,B) = \frac{\sum_{c \in LN(A,B)} coh^{*}_{...}(A,B)/coh^{*}_{...}(c)}{|LN(A,B)|}$$

$$coh^{m*}_{...,un}(A,B) = 1 - \max_{c \in UN(A,B)} coh^{*}_{...}(c)/coh^{*}_{...}(A,B)$$

$$coh^{m*}_{...,ln}(A,B) = \min_{c \in LN(A,B)} coh^{*}_{...}(A,B)/coh^{*}_{...}(c)$$

where $UN(A,B)$ and $LN(A,B)$ are upper and lower neighbors of a formal concept (A,B) respectively.

As the authors noted, experiments revealed that the type of cohesion function does not affect the result, while the choice of similarity measure can greatly affect the outcome. More than that, in some cases upper (lower) neighbors may have higher (lower) cohesion than the formal concept itself (for example, some boundary cases, when a neighbors's extent (an intent) consists of identical rows (columns) of a formal context). To tackle this issue of non-monotonic neighbors w.r.t. similarity function authors proposed to take $coh^{**}_{...,ln}$ and $coh^{**}_{...,un}$ as 0, if the rate of non-monotonic neighbors is larger that a threshold.

In our experiments we used the following notation: SMC** and J**, where the first star is replaced by a cohesion type, the second one is replaced by the type of a similarity function. Below, we consider another four metrics that were introduced in [7].

Predictability approach (P) Predictability of a formal concept is computed in a quite similar way to BL_S. A cohesion function is replaced by a predictability function:

$$P(A,B) = pred^{**}(A,B) \otimes pred^{**}_{un}(A,B) \otimes pred^{**}_{ln}(A,B)$$

The main idea behind this approach is to assign high score to concept (A,B) with low conditional entropy of the presence of attributes not in B in intents of objects from A (i.e., requiring few attributes outside B in objects from A)[7]:

$$E\left(\mathbb{I}\left[\langle x,y\rangle \in I\right]|\mathbb{I}\left[x \in A\right]\right) = -\frac{|A \cap y'|}{|A|} log \frac{|A \cap y'|}{|A|}$$

$$pred\left(A,B\right) = 1 - \sum_{y \in M-B} \frac{E\left(\mathbb{I}\left[\langle x,y\rangle \in I\right]|\mathbb{I}\left[x \in A\right]\right)}{|M-B|}.$$

Cue Validity (CV), Category Feature Collocation (CFC), Category Utility (CU) The following measures based on the conditional probability of object $g \in A$ given that $y \subseteq g'$ were introduced in [7]:

$$CV\left(A,B\right) = \sum_{y \in B} P\left(A|y'\right) = \sum_{y \in B} \frac{|A|}{|y'|}$$

$$CFC\left(A,B\right) = \sum_{y \in M} p\left(A|y'\right) p\left(y'|A\right) = \sum_{y \in M} \frac{|A \cap y'|}{|y'|} \frac{|A \cap y'|}{|A|}$$

$$CU\left(A,B\right) = p\left(A\right) \sum_{y \in M} \left[p\left(y'|A\right)^2 - p\left(y'\right)^2\right] = \frac{|A|}{|G|} \sum_{y \in M} \left[\left(\frac{|A \cap y'|}{|y'|}\right)^2 - \left(\frac{|y'|}{|G|}\right)^2\right]$$

The main intuition behind CV is to express probability of extent given attributes from intent, CFC index takes into account the relationship between all attributes of the concept and intent of the formal concept, while CU evaluates how much an attribute in an intent is characteristic for a given concept rather than for the whole context [36].

Metrics for arbitrary itemsets

Frequency(support) It is one of the most popular measures in the theory of pattern mining. According to this index the most "interesting" concepts are frequent ones (having high support). For an arbitrary formal concept the support is defined as follows

$$supp\left(A,B\right) = \frac{|A|}{|G|}$$

The support provides efficient level-wise algorithms for constructing semilattices since it exhibits anti-monotonicity (a priori property [2, 30]):

$$B_1 \subset B_2 \rightarrow supp\left(B_1\right) \geq supp\left(B_2\right)$$

Lift In the previous section different methods with background knowledge were considered. Another way to add additional knowledge to data is proposed in [11]. Under assumption of attributes independence it is possible to compute individual frequencies of attributes and take their product as the expected frequency. The ratio of the observed frequency to its expectation is defined as *lift*. The lift of a formal concept (A, B) is defined as follows:

$$lift\left(B\right) = \frac{P\left(A\right)}{\prod_{b \in B} P\left(b'\right)} = \frac{|A|/|G|}{\prod_{b \in B} |b'|/|G|}$$

Collective Strength The collective strength [1] combines ideas of comparing the observed data and expectation under the assumption of independence of attributes. To calculate this measure for a formal concept (A, B) one needs to define for B the set of objects V_B that has at least one attribute in B, but not all of them at once. Denote $q = \prod_{b \in B} supp\,(b')$ and $supp\,(V_B) = v$, the collective strength of a formal concept has the following form:

$$cs\,(B) = \frac{1-v}{v}\frac{q}{1-q}$$

4 Experiments

In this section, we compare measures with respect to their ability to help selecting most interesting concepts and filtering concepts coming from noisy datasets. For both goals, one is interested in a ranking of concepts rather than in particular values of the measures.

4.1 Formal Concept Mining

Usually concept lattices constructed from empirical data have huge amount of formal concepts, many of them being redundant, excessive and useless. In this connection the measures can be used to estimate how meaningful a concept is. Since the "interestingness" of a concept is a fairly subjective measure, the correct comparison of indices in terms of ability to select meaningful ones is impossible. With this respect we focus on similarity of indices described above. To identify how similar indices are, we use the Kendall tau correlation coefficient [22]. Put differently, we consider pairwise similarity of two lists of the same concepts that are ordered by values of the chosen indices. A set of strongly correlated measures can be replaced by one with the lowest computational complexity.

We randomly generated 100 formal contexts of random sizes. The number of attributes was in range between 10 and 40, while the number of objects varied from 10 to 70. For generated contexts we calculated pairwise Kendall tau for all indices of each context.The averaged values of correlations coefficients are represented in Table 1.

In [7] it was shown that the CU, CFC and CV are correlated, while S and P are not strongly correlated to other metrics. The results of our simulations allow us to conclude that CU, CFC and CV are also pairwise correlated to separation and support. Moreover, support is strongly correlated to separation and probability. Since the computational complexity of support is less than that of separation and probability, it is preferable to use support. It is worth noting that predictability (P) and robustness are not correlated to any other metrics and hence they can not be replaced by the metrics introduced so far.

Thus, based on the correlation analysis, it is possible to reduce computationally complexity by choosing the most easily computable index within the class of correlated metrics.

Table 1. Kendall tau correlation coefficient for indices

	S_J^{mm}	S_J^{ma}	S_J^{am}	S_J^{aa}	S_{SMC}^{mm}	S_{SMC}^{ma}	S_{SMC}^{am}	S_{SMC}^{aa}	P	CU	CFC	CV	$Rob_{0.8}$	$Rob_{0.5}$	$Rob_{0.3}$
Prob	0.18	0.15	0.14	0.14	0.04	0.03	0.00	-0.02	0.04	0.30	0.49	-0.01	-0.07	-0.11	-0.14
Sep	0.20	0.20	0.18	0.18	0.07	0.07	0.14	0.12	0.05	0.36	0.45	0.54	-0.11	-0.12	-0.13
CS	-0.08	-0.05	-0.06	-0.05	-0.07	-0.07	0.02	0.04	-0.09	0.04	-0.12	0.29	0.00	0.02	0.04
Lift	-0.16	-0.13	-0.08	-0.07	-0.09	-0.08	0.02	0.03	-0.15	-0.07	-0.25	0.25	0.07	0.10	0.11
Sup	0.17	0.17	0.21	0.21	-0.01	-0.02	0.03	0.00	-0.06	0.54	0.80	0.31	-0.10	-0.15	-0.18
Stab	0.08	0.08	0.11	0.11	0.01	0.01	-0.02	-0.02	-0.18	-0.05	0.08	0.12	0.23	0.14	0.06
$Stab_l$	0.06	0.06	0.11	0.11	0.02	0.02	0.01	0.01	-0.17	-0.16	-0.05	0.07	0.24	0.21	0.14
$Stab_h$	0.15	0.14	0.15	0.14	0.02	0.01	-0.04	-0.05	-0.11	0.24	0.45	0.23	0.13	0.00	-0.09
$Rob_{0.1}$	-0.09	-0.09	-0.02	-0.02	0.00	0.00	-0.01	0.00	-0.02	-0.11	-0.16	-0.09	0.56	0.73	0.86
$Rob_{0.3}$	-0.10	-0.10	-0.03	-0.02	0.00	0.00	-0.02	0.00	-0.03	-0.12	-0.18	-0.09	0.68	0.86	
$Rob_{0.5}$	-0.08	-0.08	-0.02	-0.02	0.02	0.02	-0.02	-0.01	-0.03	-0.12	-0.15	-0.07	0.82		
$Rob_{0.8}$	-0.06	-0.06	-0.03	-0.02	0.03	0.03	-0.03	-0.02	-0.03	-0.11	-0.12	-0.06			
CV	0.08	0.09	0.15	0.15	-0.04	-0.04	0.05	0.05	-0.14	0.50	0.52				
CFC	0.09	0.08	0.15	0.15	-0.13	-0.13	-0.05	-0.06	-0.18	0.72					
CU	0.03	0.04	0.10	0.11	-0.13	-0.13	-0.06	-0.07	-0.17					-0.11	$Stab_h$
P	0.43	0.42	0.28	0.27	0.50	0.50	0.40	0.41					0.39	0.09	$Stab_l$
S_{SMC}^{aa}	0.39	0.39	0.56	0.56	0.49	0.50	0.92					0.86	0.59	0.03	Stab
S_{SMC}^{am}	0.39	0.38	0.58	0.57	0.48	0.49					0.18	0.02	0.58	-0.17	Sup
S_{SMC}^{ma}	0.51	0.50	0.37	0.37	0.96					-0.47	-0.04	0.05	-0.29	0.10	Lift
S_{SMC}^{mm}	0.51	0.48	0.36	0.36					0.64	-0.32	-0.09	-0.04	-0.25	0.03	CS
S_J^{aa}	0.41	0.42	0.95					0.14	0.01	0.42	0.03	-0.02	0.20	-0.13	Sep
S_J^{am}	0.42	0.41					0.17	-0.53	-0.73	0.76	0.15	0.02	0.48	-0.14	Prob
S_J^{ma}	0.90						Sep	CS	Lift	Sup	Stab	$Stab_l$	$Stab_h$	$Rob_{0.1}$	

4.2 Noise Filtering

In practice, we often have to deal with noisy data. In this case, the number of formal concepts can be very large and the lattice structure becomes too complicated [24]. To test the ability to filter out noise we took 5 lattices of different structure. Four of them are quite simple (Fig. 1) and the fifth one is the binarized fragment of the Mushroom data set [1] on 500 objects and 14 attributes, its concept lattice consists of 54 formal concepts.

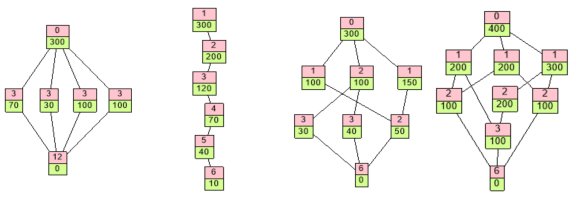


Fig. 1. Concept lattices for formal contexts with 300 objects and 6 attributes (a - c),with 400 objects and 4 attributes (d)

[1] https://archive.ics.uci.edu/ml/datasets/Mushroom

For a generated 0-1 datatable we changed table elements (0 to 1 and 1 to 0) with a given probability. The rate of noise (the probability of replacement) varied in the range from 0.05 to 0.5. We test the ability of a measure to filter redundant concepts in terms of precision and recall. For top-n (w.r.t. a measure) formal concepts, the recall and precision are defined as follows:

$$recall_{top-n} = \frac{|original \quad concepts_{top-n}|}{|original \quad concepts|}$$

$$precision_{top-n} = \frac{|original \quad concepts_{top-n}|}{|top-n \quad concepts|}$$

Table 2. Precision of indices with $recall = 0,6$

	Noise rate	Prob	Sep	Stab_l	Stab_h	CV	CFC	CU	Freq	$Rob_{0.5}$
Antichain	0.1	0.03	1	1	1	1	0.15	0.25	0.13	0.05
	0.3	0.03	1	1	1	1	0.09	0.20	0.10	0.02
	0.5	0.02	0.20	0.12	0.13	0.29	0.07	0.10	0.06	0.02
Chain	0.1	0.80	0.44	1	1	0.67	0.27	0.13	0.27	0.80
	0.3	0.21	0.18	0.67	1	0.22	0.18	0.17	0.19	1
	0.5	0.29	0.13	0.25	0.57	0.21	0.14	0.16	0.14	0.57
Context 3	0.1	0.20	1	1	1	0.36	0.33	0.44	0.67	0.40
	0.3	0.16	0.67	0.80	0.80	0.44	0.33	0.44	0.50	0.40
	0.5	0.19	0.50	0.50	0.50	0.44	0.27	0.33	0.50	0.57
Context 4	0.1	0.44	1.00	1.00	1.00	1.00	0.80	0.57	0.80	0.50
	0.3	0.22	1.00	1.00	1.00	1.00	0.80	0.57	0.80	0.57
	0.5	0.14	0.67	1.00	1.00	0.44	0.80	0.57	0.80	0.67
Mushroom	0.1	0.28	0.29	0.84	0.84	0.32	0.28	0.32	0.31	0.30
	0.3	0.16	0.16	0.36	0.39	0.25	0.18	0.20	0.22	0.09
	0.5	0.08	0.10	0.17	0.17	0.14	0.11	0.16	0.11	0.06

Figures 2 show the ROC curve for the measures. The curves that are close to the left upper corner correspond to the most powerful measures.

The best and most stable results correspond to the high estimate of stability (stability_h). The similar precision has the lower estimate of stability (Table 2), whereas precision of separation and probability depends on the proportion of noise and lattice structure as well. The measures of basic level that utilize similarity and predictability approaches become zero for some concepts. The rate of vanished concepts (including original ones) increases as the noise probability gets bigger. In our study we take such concepts as "false negative", so in this case ROC curves do not pass through the point (1,1). More than that, recall and

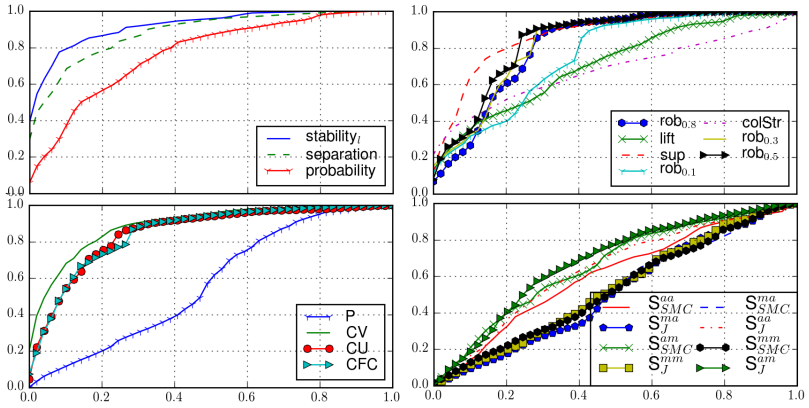


Fig. 2. Averaged ROC curves of indices among contexts 1 - 5 with different noise rate (0.1 - 0.5)

precision are unstable with respect to the noise rate and lattice structure. This group of measures is inappropriate for noise filtering.

The other basic level measures, such as CU, CFC and CV, demonstrate much better recall compared to previous ones. However, in general the precision of CU, CFC and CV is determined by lattice structure (Table 2).

Frequency has the highest precision among the indices that are applicable for the assessment of arbitrary sets of attributes. Frequency is stable with respect to the noise rate, but can vary under different lattice structures. For the lift and the collective strength precision depends on the lattice structure, and the collective strength also has quite unstable recall.

Precision of robustness depends on both lattice structure and value of α (Fig. 2). In our study we have got the highest precision for α close to 0.5.

Thus, the most preferred metrics for noise filtering are stability estimates, CV, frequency and robustness (where α is greater than 0.4).

In [24] it was noticed that the combination of the indices can improve the filtering power of indices. In this regard, we have studied top-n concepts selected by pairwise combination of measures. As it was shown by the experiments, the combination of measures may improve recall of the top-n set, while precision gets lower with respect to a more accurate measure. Figure 3 shows recall and precision of different combination of measures. In the best case it is possible to improve the recall, the precision on small sets of top-n concepts is lower than the precision of one measure by itself.

5 Conclusion

In this paper we have considered various methods for selecting interesting concepts and noise reduction. We focused on the most promising and well interpretable approach based on interestingness measures of concepts. Since “inter-

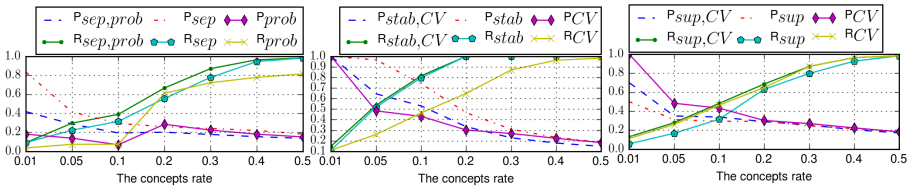


Fig. 3. Recall and precision of metrics and their combination on a Mushroom dataset fragment with the noise probability 0.1

estingness" of a concept is a subjective measure, we have compared several measures known in the literature and identified groups of most correlated ones. CU, CFC, CV, separation and frequency make up the first group. Frequency is correlated to separation and probability.

Another part of our experiments was focused on the noise filtering. We have found that the stability estimates work perfectly with data of various noise rate and different structure of the original lattice. Robustness and 3 of basic level metrics (cue validity, category utility and category feature collocation approaches) could also be applied to noise reduction. The combination of measures can also improve the recall, but only in the case of high noise rate.

Acknowledgments

The authors were supported by the project "Mathematical Models, Algorithms, and Software Tools for Mining of Structural and Textual Data" supported by the Basic Research Program of the National Research University Higher School of Economics.

References

1. Aggarwal, C.C., Yu, P.S.: A new framework for itemset generation. In: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. pp. 18–24. ACM (1998)
2. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)
3. Arévalo, G., Berry, A., Huchard, M., Perrot, G., Sigayret, A.: Performances of galois sub-hierarchy-building algorithms. In: Formal Concept Analysis, pp. 166–180. Springer (2007)
4. Babin, M.A., Kuznetsov, S.O.: Approximating concept stability. In: Domenach, F., Ignatov, D., Poelmans, J. (eds.) Formal Concept Analysis. Lecture Notes in Computer Science, vol. 7278, pp. 7–15. Springer Berlin Heidelberg (2012)
5. Belohlavek, R., Macko, J.: Selecting important concepts using weights. In: Valtchev, P., Jschke, R. (eds.) Formal Concept Analysis, Lecture Notes in Computer Science, vol. 6628, pp. 65–80. Springer Berlin Heidelberg (2011)
6. Belohlavek, R., Trnecka, M.: Basic level of concepts in formal concept analysis. In: Domenach, F., Ignatov, D., Poelmans, J. (eds.) Formal Concept Analysis, Lecture Notes in Computer Science, vol. 7278, pp. 28–44. Springer Berlin Heidelberg (2012)

7. Belohlavek, R., Trnecka, M.: Basic level in formal concept analysis: Interesting concepts and psychological ramifications. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. pp. 1233–1239. IJCAI '13, AAAI Press (2013)
8. Belohlavek, R., Vychodil, V.: Formal concept analysis with background knowledge: attribute priorities. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 39(4), 399–409 (2009)
9. Berry, A., Huchard, M., McConnell, R., Sigayret, A., Spinrad, J.: Efficiently computing a linear extension of the sub-hierarchy of a concept lattice. In: Ganter, B., Godin, R. (eds.) Formal Concept Analysis, Lecture Notes in Computer Science, vol. 3403, pp. 208–222. Springer Berlin Heidelberg (2005)
10. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Scalable estimates of concept stability. In: Glodeanu, C., Kaytoue, M., Sacarea, C. (eds.) Formal Concept Analysis, Lecture Notes in Computer Science, vol. 8478, pp. 157–172. Springer International Publishing (2014)
11. Cabena, P., Choi, H.H., Kim, I.S., Otsuka, S., Reinschmidt, J., Saarenvirta, G.: Intelligent miner for data applications guide. IBM RedBook SG24-5252-00 173 (1999)
12. Carpineto, C., Romano, G.: A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning 24(2), 95–122 (1996)
13. Cheung, K., Vogel, D.: Complexity reduction in lattice-based information retrieval. Information Retrieval 8(2), 285–299 (2005)
14. Dhillon, I., Modha, D.: Concept decompositions for large sparse text data using clustering. Machine Learning 42(1-2), 143–175 (2001)
15. Dias, S.M., Vieira, N.: Reducing the size of concept lattices: The JBOS approach. In: Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, October 19-21, 2010. pp. 80–91 (2010)
16. Dias, S.M., Vieira, N.J.: Concept lattices reduction: Definition, analysis and classification. Expert Systems with Applications 42(20), 7084 – 7097 (2015)
17. Dobša, J., Dalbelo-Bašić, B.: Comparison of information retrieval techniques: latent semantic indexing and concept indexing. Journal of Inf. and Organizational Sciences 28(1-2), 1–17 (2004)
18. Düntsch, I., Gediga, G.: Simplifying contextual structures. In: Kryszkiewicz, M., Bandyopadhyay, S., Rybinski, H., Pal, S.K. (eds.) Pattern Recognition and Machine Intelligence, Lecture Notes in Computer Science, vol. 9124, pp. 23–32. Springer International Publishing (2015)
19. Emilion, R.: Concepts of a discrete random variable. In: Brito, P., Cucumel, G., Bertrand, P., de Carvalho, F. (eds.) Selected Contributions in Data Analysis and Classification, pp. 247–258. Studies in Classification, Data Analysis, and Knowledge Organization, Springer Berlin Heidelberg (2007)
20. Ganter, B., Wille, R.: Contextual attribute logic. In: Tepfenhart, W., Cyre, W. (eds.) Conceptual Structures: Standards and Practices, Lecture Notes in Computer Science, vol. 1640, pp. 377–388. Springer Berlin Heidelberg (1999)
21. Jay, N., Kohler, F., Napoli, A.: Analysis of social communities with iceberg and stability-based concept lattices. In: Medina, R., Obiedkov, S. (eds.) Formal Concept Analysis, Lecture Notes in Computer Science, vol. 4933, pp. 258–272. Springer Berlin Heidelberg (2008)
22. Kendall, M.G.: A new measure of rank correlation. Biometrika pp. 81–93 (1938)
23. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms. Springer Netherlands (2000)

24. Klimushkin, M., Obiedkov, S., Roth, C.: Approaches to the selection of relevant concepts in the case of noisy data. In: Kwuida, L., Sertkaya, B. (eds.) Formal Concept Analysis, Lecture Notes in Computer Science, vol. 5986, pp. 255–266. Springer Berlin Heidelberg (2010)
25. Kumar, C.A., Srinivas, S.: Latent semantic indexing using eigenvalue analysis for efficient information retrieval. Int. J. Appl. Math. Comput. Sci 16(4), 551–558 (2006)
26. Kuznetsov, S.O.: Interpretation on graphs and complexity characteristics of a search for specific patterns. Automatic Documentation and Mathematical Linguistics 24(1), 37–45 (1989)
27. Kuznetsov, S.O.: Stability as an estimate of degree of substantiation of hypotheses derived on the basis of operational similarity. Nauchn. Tekh. Inf., Ser. 2 (12), 21–29 (1990)
28. Kuznetsov, S.O.: On stability of a formal concept. Annals of Mathematics and Artificial Intelligence 49(1-4), 101–115 (2007)
29. Kuznetsov, S.O., Obiedkov, S., Roth, C.: Reducing the representation complexity of lattice-based taxonomies. In: Conceptual Structures: Knowledge Architectures for Smart Applications, pp. 241–254. Springer Berlin Heidelberg (2007)
30. Mannila, H., Toivonen, H., Verkamo, A.I.: Efficient algorithms for discovering association rules. In: KDD-94: AAAI workshop on Knowledge Discovery in Databases. pp. 181–192 (1994)
31. Murphy, G.L.: The big book of concepts. MIT press (2002)
32. Rosch, E.: Principles of categorization pp. 27–48 (1978)
33. Snasel, V., Polovincak, M., Abdulla, H.M.D., Horak, Z.: On concept lattices and implication bases from reduced contexts. In: ICCS. pp. 83–90 (2008)
34. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with titanic. Data Knowl. Eng. 42(2), 189–222 (Aug 2002)
35. Tatti, N., Moerchen, F., Calders, T.: Finding robust itemsets under subsampling. ACM Transactions on Database Systems (TODS) 39(3), 20 (2014)
36. Zeigenfuse, M.D., Lee, M.D.: A comparison of three measures of the association between a feature and a concept. In: Proceedings of the 33rd Annual Conference of the Cognitive Science Society. pp. 243–248 (2011)

Why concept lattices are large

Extremal theory for the number of minimal generators and formal concepts

Alexandre Albano[1] and Bogdan Chornomaz[2]

[1] Technische Universität Dresden
[2] V.N. Karazin Kharkiv National University

Abstract. A unique type of subcontexts is always present in formal contexts with many concepts: the contranominal scales. We make this precise by giving an upper bound for the number of minimal generators (and thereby for the number of concepts) of contexts without contranominal scales larger than a given size. Extremal contexts are constructed which meet this bound exactly. They are completely classified.

1 Introduction

The primitive data model of Formal Concept Analysis is that of a formal context, which is unfolded into a concept lattice for further analysis. It is well known that concept lattices may be exponentially larger than the contexts which gave rise to them. An obvious example is the boolean lattice $\mathrm{B}(k)$, having 2^k elements, the standard context of which is the $k \times k$ *contranominal scale* $\mathbb{N}^c(k)$. This is not the only example of contexts having large associated concept lattices: indeed, the lattice of any subcontext is embeddable in the lattice of the whole context [4], which means that contexts having large contranominal scales as subcontexts necessarily have large concept lattices as well. Those considerations induce one natural question, namely, whether there are other reasons for a concept lattice to be large. As it will be shown in this paper, the answer is no.

The structure of the paper is as follows. Our starting point is a known upper bound for the number of concepts, which we improve using the language of minimal generators. Then, we show that our result is the best possible by constructing lattices which attain exactly the improved upper bound. These lattices, i.e., the extremal lattices, are characterized.

2 Fundamentals

For a set S, a context of the form $(S, S, \neq)$ will be called a *contranominal scale*. We will denote by $\mathbb{N}^c(k)$ the contranominal scale with k objects (and k attributes), that is, the context $([k], [k], \neq)$, where $[k] := \{1, 2, \ldots, k\}$. The expression $\mathbb{K}_1 \leq \mathbb{K}$ denotes that $\mathbb{K}_1$ is a subcontext of $\mathbb{K}$. The symbol $\cong$ expresses the existence of an order-isomorphism whenever two ordered sets are involved

or, alternatively, the existence of a context isomorphism in the case of formal contexts. For a context $\mathbb{K}$ to be $\mathbb{N}^c(k)$-*free* means that there does not exist a subcontext $\mathbb{K}_1 \leq \mathbb{K}$ with $\mathbb{K}_1 \cong \mathbb{N}^c(k)$. The boolean lattice with k atoms, that is, $\underline{\mathfrak{B}}(\mathbb{N}^c(k))$, will be denoted by $\mathrm{B}(k)$. Similarly, we say that a lattice L is $\mathrm{B}(k)$-*free* whenever $\mathrm{B}(k)$ does not (order-)embed into L. Using Proposition 32 from [4] one has that $\mathbb{K}$ is $\mathbb{N}^c(k)$-free whenever $\underline{\mathfrak{B}}(\mathbb{K})$ is $\mathrm{B}(k)$-free. The converse is also true and is, in fact, the content of our first proposition. An example of a context which has $\mathbb{N}^c(3)$ as a subcontext along with its concept lattice is depicted in Figure 1. One may observe that the context is $\mathbb{N}^c(4)$-free, since its lattice has ten concepts (and would have at least sixteen otherwise).

	m	n	o	p	q
g		×	×		
h	×		×	×	
i	×	×		×	
j	×			×	×
k	×				

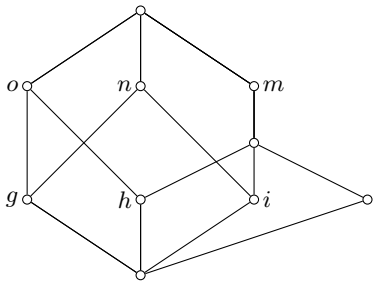


Fig. 1: A context $\mathbb{K}$ with $\mathbb{N}^c(3) \leq \mathbb{K}$ and its concept lattice. The object and attribute concepts belonging to the B(3) suborder are indicated on the diagram.

We denote by $J(L)$ and $M(L)$, respectively, the set of completely join-irreducible and meet-irreducible elements of a lattice L. The least and greatest elements of a lattice L will be denoted, respectively, by 0_L and 1_L. The symbol $\prec$ will designate the covering relation between elements, that is, $x \prec y$ if $x < y$ and for every $z \in L$, $x < z \leq y \Rightarrow z = y$. The *length* of a finite lattice is the number of elements in a maximum chain minus one. An *atom* is an element covering 0_L, while a *coatom* is an element covered by 1_L. Whenever two elements $x, y \in L$ are incomparable, we will write $x||y$. We denote by $A(L)$ the set of atoms of a lattice L. For an element $l \in L$, we shall write $\downarrow l := \{x \in L \mid x \leq l\}$ as well as $\uparrow l := \{x \in L \mid x \geq l\}$. Moreover, for $l \in L$ we denote by A_l the set $A(L) \cap \downarrow l$ and, similarly, by J_l the set $J(L) \cap \downarrow l$. A complete lattice L is called *atomistic* if $x = \bigvee A_x$ holds for every $x \in L$. In this case, $A(L) = J(L)$.

Proposition 1. *Let $\mathbb{K}$ be a context such that $\mathrm{B}(k)$ embeds into $\underline{\mathfrak{B}}(\mathbb{K})$. Then $\mathbb{N}^c(k) \leq \mathbb{K}$.*

Proof. Let $(A_1, B_1), \ldots, (A_k, B_k)$ be the atoms of $\mathrm{B}(k)$ in $\underline{\mathfrak{B}}(\mathbb{K})$. Similarly, denote its coatoms by $(C_1, D_1), \ldots, (C_k, D_k)$ in such a way that $(A_i, B_i) \leq (C_j, D_j) \Leftrightarrow i \neq j$ for each i, j. Note that the sets A_i, as well as the sets D_i, are non-empty. Let $i \in [k]$. Since $(A_i, B_i) \not\leq (C_i, D_i)$, we may take an object/attribute pair $g_i \in A_i, m_i \in D_i$ with $g_i \not\mathrel{I} m_i$. For every chosen object $g_i \in A_i$, one has that

$g_i I m_j$ for every $j \in [k]$ with $j \neq i$, because of $(A_i, B_i) \leq (C_j, D_j)$, which implies $B_i \supseteq D_j$. Consequently, k distinct objects g_i (as well as k distinct attributes m_i) were chosen. Combining both relations results in $g_i I m_j \Leftrightarrow i \neq j$ for each $i \in [k]$, that is, the objects and attributes g_i, m_i form a contranominal scale in $\mathbb{K}$. □

Definition 1. *Let (G, M, I) be a formal context. A set $S \subseteq G$ is said to be a **minimal generator** (of the extent S'') if $T'' \neq S''$ for every proper subset $T \subsetneq S$. The set of all minimal generators of a context $\mathbb{K}$ will be denoted by* MinGen($\mathbb{K}$).

Observation: In contexts with finitely many objects, every extent has at least one minimal generator. Clearly, two different extents cannot share one same minimal generator. Thus, the upper bound $|\mathfrak{B}(\mathbb{K})| \leq |\text{MinGen}(\mathbb{K})|$ holds for contexts with finite object sets.

The problem of computing exactly the number of concepts does not admit a polynomial-time algorithm, unless P=NP. This was shown by Kuznetsov [5]: more precisely, this counting problem is #P-complete. However, there are results which establish upper bounds for the number of concepts: see for example [1–3, 6, 7].

3 The upper bound

Our investigations were inspired by a result of Prisner, who gave the first upper bound regarding contranominal-scale free contexts. The original version is in graph theoretic language. Reformulated it reads as follows:

Theorem 1 (Prisner [6]). *Let $\mathbb{K} = (G, M, I)$ be a $\mathbb{N}^c(k)$-free context. Then,*

$$|\mathfrak{B}(\mathbb{K})| \leq (|G||M|)^{k-1} + 1.$$

In this section we will show an improvement of Theorem 1. For that, we will relate minimal generators with contranominal scales. The first step towards this is the equivalence shown in Proposition 2. Note that, since derivation operators are antitone, the $\neq$ symbol may be substituted by $\supsetneq$.

Proposition 2. *Let (G, M, I) be a formal context. A set $S \subseteq G$ is a minimal generator if and only if for every $g \in S$, it holds that $(S \setminus \{g\})' \neq S'$.*

Proof. We will show the two equivalent contrapositions. If $(S \setminus \{g\})' = S'$, then, of course, $(S \setminus \{g\})'' = S''$, and S is not a minimal generator. For the converse, suppose that S is not a minimal generator, and take a proper subset T of S with $T'' = S''$. Note that $T'' = S''$ implies $T' = S'$. Let $g \in S \setminus T$. On one hand, $(S \setminus \{g\}) \subseteq S$ implies $(S \setminus \{g\})' \supseteq S'$. On the other hand, $(S \setminus \{g\}) \supseteq T$ implies $(S \setminus \{g\})' \subseteq T' = S'$. Combining both yields $(S \setminus \{g\})' = S'$. □

The next proposition relates minimal generators and contranominal scales.

Lemma 1. *Let $\mathbb{K} = (G, M, I)$ be a context and $A \subseteq G$. There exists a contranominal scale $\mathbb{K}_1 \leq \mathbb{K}$ having A as its object set if and only if A is a minimal generator. In particular, if G is finite:*

$$\max_{A \subseteq G}\{|A| : A \text{ is a minimal generator}\} = \max\{k \in \mathbb{N} : \mathbb{N}^c(k) \leq \mathbb{K}\}.$$

Proof. Suppose that A is a minimal generator and let $g \in A$. By Proposition 2, one has that $(A \setminus \{g\})' \supsetneq A'$. Hence, there exists an attribute m with $g \not I m$ and hIm for every $h \in A \setminus \{g\}$. Clearly, two different objects $g_1, g_2 \in A$ cannot give rise to the same attribute m, since the two pairs of conditions $g_i \not I m$ and hIm for every $h \in A \setminus \{g_i\}$ cannot be satisfied simultaneously ($i = 1, 2$). Thus, there exists an injection $\iota : A \to M$ with $g \not I \iota(g)$, $hI\iota(g)$ for each $g \in A$ and each $h \in A \setminus \{g\}$. By setting $N = \iota(A)$, one has that $(A, N, I \cap (A \times N))$ is a contranominal scale. For the converse, let $\mathbb{K}_1 = (S, S, \neq) \leq \mathbb{K}$ be a contranominal scale and let $g \in S$. Clearly, $g \notin S'$. Moreover, $g \in (S \setminus \{g\})'$. This amounts to $(S \setminus \{g\})' \supsetneq S'$ for each $g \in S$. By Proposition 2, the set S is a minimal generator. □

A consequence of Lemma 1 is the following, which is an improvement of the order of $k! \cdot |M|^k/k$ of Prisner's bound.

Theorem 2. *Let $\mathbb{K} = (G, M, I)$ be a $\mathbb{N}^c(k)$-free context with finite G. Then:*

$$|\mathfrak{B}(\mathbb{K})| \;\leq\; |\textsc{MinGen}(\mathbb{K})| \;\leq\; \sum_{i=0}^{k-1} \binom{|G|}{i}.$$

In particular, if $k \leq \frac{|G|}{2}$:

$$|\mathfrak{B}(\mathbb{K})| \leq k \cdot \frac{|G|^{k-1}}{(k-1)!}.$$

Proof. Lemma 1 guarantees that $\mathbb{K}$ does not have any minimal generator of cardinality greater or equal to k. The sum above is the number of subsets of G having cardinality at most $k - 1$. □

Definition 2. *We denote by $f(n, k)$ the upper bound in Theorem 2:*

$$f(n, k) := \sum_{i=0}^{k-1} \binom{n}{i}.$$

The upper bound in Theorem 2 for $f(n, k)$ gets worse as k gets close to $\frac{|G|}{2}$. Tighter upper bounds for the sum of binomial coefficients may be found in [9].

4 Sharpness: preparatory results

The following property of $f(n, k)$ is needed for the next two sections.

Proposition 3. *The function $f(n,k)$ satisfies the following identity:*

$$f(n,k) = f(n-1,k-1) + f(n-1,k).$$

Proof. This follows from a standard binomial identity: $f(n-1,k) + f(n-1,k-1) = \sum_{i=0}^{k-1} \binom{n-1}{i} + \sum_{j=0}^{k-2} \binom{n-1}{j} = 1 + \sum_{i=1}^{k-1} \binom{n-1}{i-1} + \sum_{j=1}^{k-1} \binom{n-1}{j} = 1 + \sum_{i=1}^{k-1} \binom{n}{i} = f(n,k)$. □

Consider a finite lattice L. It is well known that every element $x \in L$ is the supremum of some subset of $J(L)$: for example, $x = \bigvee J_x$. We call such a subset a *representation of x through join-irreducible elements* (for brevity, we may say *a representation through irreducibles of x* or even only a *representation of x*). A representation $S \subseteq J(L)$ of x is called *irredundant* if $\bigvee(S \setminus \{y\}) \neq x$ for every $y \in S$. Of course, every $x \in L$ has an irredundant representation, but it does not need to be unique. Note that irredundant representations are precisely minimal generators when one takes the standard context of L, $(J(L), M(L), \leq)$. Indeed, in that formal context, the closure of object sets corresponds to the supremum of join-irreducible elements of L. For an element $x \in L$, there may exist elements in J_x which belong to every representation of x: the so-called *extremal points*. An element $z \in J_x$ is an *extremal point of x* if there exists a lower neighbor y of x such that $J_y = J_x \setminus \{z\}$. Every representation of x must contain every extremal point z of x since, in this case, the supremum $\bigvee(J_x \setminus \{z\})$ is strictly smaller than x (and is actually covered by x).

In Section 5 we shall construct finite lattices for which every element has exactly one irredundant representation. It turns out that, in the finite case, these lattices are precisely the meet-distributive lattices. This is implied by Theorem 44 of [4], which actually gives information about the unique irredundant representation as well: a finite lattice L is meet-distributive if and only if for every $x \in L$ the set E_x of all extremal points of x is a representation of x (and E_x is, therefore, the unique irredundant representation of x, since every representation of x must contain E_x). Proposition 4 provides a characteristic property for the finite case which will be used in our constructions.

Proposition 4. *Let L be a finite lattice. The following assertions are equivalent:*

i) L is meet-distributive.
ii) Every element $x \in L$ is the supremum of its extremal points.
iii) For every $x, y \in L$ with $x \prec y$, it holds that $|J_y \setminus J_x| = 1$.

Proof. The equivalence between *i*) and *ii*) may be found in Theorem 44 of [4]. Let $x \in L$ and define $E_x = \{z \in J_x \mid z \text{ is an extremal point of } x\}$. We now show that *ii*) implies *iii*). Let $y \in L$ with $y < x$. This implies $J_y \subsetneq J_x$. The set J_y does not contain E_x, because this would force $y \geq x$. Therefore, $y = \bigvee J_y$ is upper bounded by some element in the set $U = \{\bigvee(J_x \setminus \{z\}) \mid z \in E_x\}$ (note that $x \notin U$). Hence, every lower neighbor of x has a representation of the form $(J_x \setminus \{z\})$ with $z \in E_x$. Now we show that *iii*) implies *ii*). Define $y = \bigvee E_x$ and suppose by contradiction that $y < x$. Then, there exists an element z such

that $y \leq z \prec x$ and $J_z \supseteq E_x$. But then, $z \prec x$ implies $J_x \setminus J_z = \{w\}$ for some $w \in J(L)$, which means that w is an extremal point of x. This contradicts the fact that E_x contains all extremal points of x. □

The next lemma will be useful in Section 5, when we shall change the perspective from lattices to contexts.

Lemma 2. *Let L be a finite lattice. If L is $\mathrm{B}(k)$-free, then every element has a representation through join-irreducibles of size at most $k-1$. The converse holds if L is meet-distributive.*

Proof. Let $\mathbb{K} = (J(L), M(L), \leq)$ be the standard context of L. We identify the elements of L with the extents of $\mathbb{K}$ via $x \mapsto J_x$. Suppose that L is $\mathrm{B}(k)$-free. Then, $\mathbb{K}$ is $\mathbb{N}^c(k)$-free. Let A be an arbitrary extent of $\mathbb{K}$ and S a minimal generator of A. Then, by Lemma 1, it follows that $|S| \leq k-1$. Since $A = S'' = \bigvee S$, we have the desired representation. Now, suppose that $|J_y \setminus J_x| = 1$ holds for every $x, y \in L$ with $x \prec y$ (cf. Proposition 4). To prove the converse, we suppose that $\mathrm{B}(k)$ embeds into L and our goal is to show that some $x \in L$ does not have any representation with fewer than k elements of $J(L)$. Now, since $\mathrm{B}(k)$ embeds into L, Proposition 1 implies that $\mathbb{N}^c(k)$ is a subcontext of $\mathbb{K}$. Applying Lemma 1, we have that there exists a minimal generator $S \subseteq J(L)$ with $|S| = k$. Equivalently, S is an irredundant representation of the element S'' of L. By Proposition 4, S is the unique irredundant representation of S''. Therefore, S'' cannot be expressed as the supremum of fewer than k join-irreducible elements. □

5 Sharpness: construction of extremal lattices

In this section, we will consider only finite lattices. Our objective is to construct lattices which prove that the bound in Theorem 2 is sharp.

Definition 3. *For positive integers n and k, we call a lattice **(n,k)-extremal** if it has at most n join-irreducible elements, is $\mathrm{B}(k)$-free, and has exactly $f(n,k)$ elements.*

It is clear that every $(n,1)$-extremal lattice is trivial, i.e., the lattice with one element. To construct (n,k)-extremal lattices with larger k, we will use an operation which we call *doubling*.

Definition 4. *Let L be an ordered set and $K \subseteq L$. The **doubling of K in L** is defined to be $L[K] = L \cup \dot{K}$, where $\dot{K}$ is a disjoint copy of K, i.e., $\dot{K} \cap L = \emptyset$. The order in $(L[K], \leq')$ is defined as follows:*

$$\leq' = \leq \cup \{(x, \dot{y}) \in L \times \dot{K} \mid x \leq y\} \cup \{(\dot{x}, \dot{y}) \in \dot{K} \times \dot{K} \mid x \leq y\}.$$

We will employ the notation $\dot{x}$ to denote the image under doubling of an element $x \in K$. Note that $x \prec \dot{x}$ for every $x \in K$, and that $\dot{x}$ is the only upper neighbor of x in $\dot{K}$. When L is a set family $\mathcal{C} \subseteq \mathcal{P}(G)$, then the diagram of $L[K]$ can be easily depicted: the doubling $\mathcal{C}[\mathcal{D}]$ (with $\mathcal{D} \subseteq \mathcal{C}$) corresponds to the set family $\mathcal{C} \cup \{D \cup \{g\} \mid D \in \mathcal{D}\}$, where $g \notin G$ is a new element. Figure 2 illustrates three doubling operations. The first one is the doubling of the chain $\{\emptyset, \{2\}, \{1,2\}\}$ inside the closure system $\mathcal{C}_1 = \mathcal{P}([2])$, resulting in $\mathcal{C}_2$. The (*a fortiori*) closure systems $\mathcal{C}_3$ and $\mathcal{C}_4$ are obtained by doubling, respectively, the chains $\{\emptyset, \{3\}, \{2,3\}, \{1,2,3\}\}$ and $\{\emptyset, \{2\}, \{2,3\}, \{1,2,3\}\}$ inside $\mathcal{C}_2$.

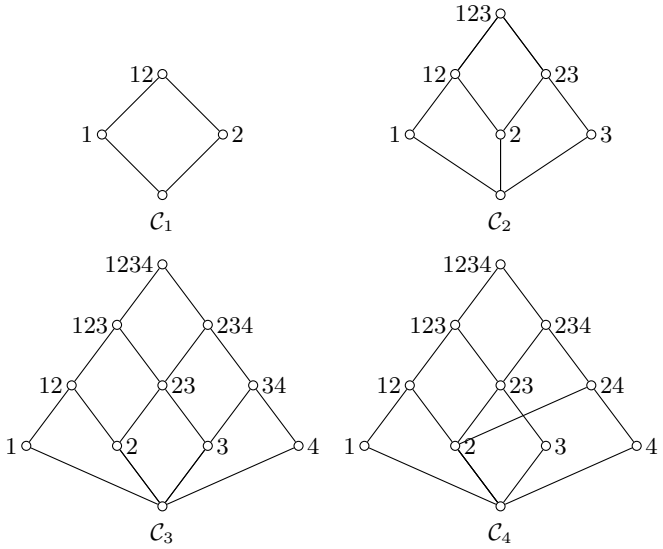


Fig. 2: Doubling chains inside closure systems

Since we are interested in constructing lattices, it is vital to guarantee that the doubling operation produces a lattice. By a *meet-subsemilattice* of a lattice L is meant a subset K of L, endowed with the inherited order, such that $x \wedge y \in K$ holds for every $x, y \in K$. It is called *topped* if $1_L \in K$.

Proposition 5. *If K is a topped meet-subsemilattice of a lattice L, then $L[K]$ is a lattice.*

Proof. Let $x, y \in L[K]$. If both x and y belong to L, then clearly $x \wedge y$ and $x \vee y$ belong to $L \subseteq L[K]$. Suppose that only one among x and y, say x, belongs to L. Then $y = \dot{z}$ with $z \in K$. We have that $x \wedge y = x \wedge z \in L \subseteq L[K]$ because of $x \not\leq \dot{0}_K$ and $y = z \vee \dot{0}_K$. For the supremum, set $S = \{w \in K \mid w \geq x, w \geq z\}$ and $u = \bigwedge S$. Note that the fact that K is topped causes $S \neq \emptyset$. Since K is a meet-subsemilattice, we have that $u \in K$. It is clear that u is the least upper

bound of x and z which belongs to K. Therefore, $\dot{u}$ is the least upper bound of x and y, because of $0_{\dot{K}} \not\leq u$ and $y = z \vee 0_{\dot{K}}$. The remaining case is $x, y \in \dot{K}$ for which, clearly $x \wedge y$ exists. Moreover, writing $x = \dot{t}, y = \dot{z}$ with $t, z \in K$ and setting $S = \{w \in K \mid w \geq t, w \geq z\}$ as well as $u = \bigwedge S$ make clear that $\dot{u} = x \vee y$. □

When extrinsically considered, topped meet-subsemilattices are lattices. This is compatible with the proof of Proposition 5, where the supremum and infimum of two elements in $\dot{K}$ may be easily verified to belong to $\dot{K}$: that is, $\dot{K}$ is actually a sublattice of $L[K]$.

A suborder K of an ordered set L is called *cover-preserving* if $x \prec_K y$ implies $x \prec_L y$ for every $x, y \in K$. This property plays a key role by preserving meet-distributivity under a doubling operation:

Proposition 6. *Let L be a meet-distributive lattice and let K be a cover-preserving, topped meet-subsemilattice of L. Then, $L[K]$ is a meet-distributive lattice.*

Proof. The fact that $L[K]$ is a lattice comes from Proposition 5. Every element $\dot{x} \in \dot{K}$ has one lower neighbor in K, namely, x. Thus, the total number of lower neighbors of $\dot{x}$ is one only if x does not cover any element in K, that is, $x = 0_K$. Therefore, $0_{\dot{K}}$ is the only join-irreducible of $L[K]$ which is not a join-irreducible of L. Let $x, y \in L[K]$ with $x \prec_{L[K]} y$. We use $J'_{(\cdot)}$ to denote our J-notation in $L[K]$ and $J_{(\cdot)}$ in L. If $x, y \in L$, then clearly $J'_y = J_y$ and $J'_x = J_x$, which results in $|J'_y \setminus J'_x| = |J_y \setminus J_x| = 1$. If $x, y \notin L$, then $x = \dot{z}$ and $y = \dot{w}$ with $z, w \in K$ and $z \prec_K w$. From the fact that K is cover-preserving, we conclude that $z \prec_L w$. Because L is meet-distributive, it follows that $|J_w \setminus J_z| = 1$. Clearly one has $J'_x = J_z \cup \{0_{\dot{K}}\}$ and $J'_y = J_w \cup \{0_{\dot{K}}\}$, which yields $|J'_y \setminus J'_x| = 1$. For the remaining case, one has necessarily $x \in L$ and $y \notin L$. In these conditions, $x \prec y$ results in $y = \dot{x}$ and, therefore, $J'_y = J'_x \cup \{0_{\dot{K}}\}$, implying $|J'_y \setminus J'_x| = 1$. □

Proposition 7 is the first assertion about extremal meet-subsemilattices. We note that the set of join-irreducible elements of a meet-subsemilattice K of a lattice L is not the same as the set $J(L) \cap K$. Therefore, what is meant by an (n, k)-*extremal meet-subsemilattice of* L is precisely the following: a lattice K which is (n, k)-extremal and a meet-subsemilattice of L as well.

Observe that chains with $n + 1$ elements are precisely the $(n, 2)$-extremal lattices. Proposition 7 illustrates, in particular, that an $n+1$ element chain may be seen as the result of a doubling operation on an n element chain, provided that the doubling operation is performed with respect to the trivial topped meet-subsemilattice $\uparrow 1$, which is $(n, 1)$-extremal.

Proposition 7. *Let L be an $(n-1, k)$-extremal lattice with $n, k \geq 2$. Suppose that K is a topped, $(n-1, k-1)$-extremal meet-subsemilattice. If $L[K]$ is* B(k)*-free, then it is an (n, k)-extremal lattice.*

Proof. Proposition 5 guarantees that $L[K]$ is indeed a lattice. As in the proof of Proposition 6, we have that $J(L[K]) = J(L) \cup \{\dot{0}_K\}$ and in particular, $L[K]$ has at most n join-irreducible elements. The claim that $L[K]$ has $f(n,k)$ elements follows from Proposition 3. □

It is clear now how $(n,2)$-extremal lattices can be obtained by doubling a trivial meet-subsemilattice of an $(n-1,2)$-extremal lattice. The succeeding propositions and lemmas aim particularly towards a generalization of this operation: the doubling of topped, $(n-1,k-1)$-extremal meet-subsemilattices inside $(n-1,k)$-extremal lattices, yielding (n,k)-extremal lattices for $k \geq 3$.

Proposition 8. *Suppose that L is an (n,k)-extremal lattice. Then, for every $S,T \subseteq J(L)$ with $|S|,|T| \leq k-1$:*

$$\bigvee S = \bigvee T \Rightarrow S = T.$$

Moreover, if $k \geq 2$ then $|J(L)| = n$.

Proof. We may suppose $k \geq 2$ since the assertion holds trivially for $k = 1$. Lemma 2 guarantees that every element x of L has a representation of size at most $k-1$. Therefore $L = \{\bigvee S \mid S \subseteq J(L), |S| \leq k-1\}$. Because of $k \geq 2$ and the fact that $|L| = f(n,k)$ is also the number of subsets of $[n]$ having at most $k-1$ elements, one has $|J(L)| \geq n$. In fact equality must hold, because L has at most n join-irreducible elements. As a consequence of $|J(L)| = n$ and $|L| = f(n,k)$, we have that no two sets $S,T \subseteq J(L)$ with $S \neq T$ may lead to the same supremum $\bigvee S = \bigvee T$. □

Chains are the only extremal lattices which are not atomistic, as a consequence of the next lemma.

Lemma 3. *Suppose that L is an (n,k)-extremal lattice with $k \geq 3$. Then L is atomistic and meet-distributive. In particular, the length of L equals the number of its atoms and there exists an atom which is an extremal point of 1_L.*

Proof. If L were not atomistic, there would exist two comparable join-irreducible elements, say, x,y with $x < y$. But then $x \vee y = y$, which contradicts Proposition 8. Suppose that L is not meet-distributive and take $x,y \in L$ with $x \prec y$ such that $A_y \setminus A_x$ has at least two elements. Clearly $x \neq 0_L$ and, therefore, $A_x \neq \emptyset$. Let $u,v \in A_y \setminus A_x$ be any two distinct elements. From $u \notin A_x$ follows that $x < x \vee u \leq y$ which, in turn, implies $x \vee u = y$. Similarly, $v \notin A_x$ implies $x < x \vee v \leq y$ which, in turn, implies $x \vee v = y$. Let $a \in A_x$. Now, $a \leq x$ and $x||u$ imply $a \vee u = x \vee u = y$, as well as $a \leq x$ and $x||v$ imply $a \vee v = x \vee v = y$. We obtain $a \vee u = a \vee v$, contradicting Proposition 8. Choosing a maximal chain $x_0 \prec x_1 \prec \ldots \prec x_l$ in L and noticing that the sizes of the sets A_{x_i} grow by exactly one element make the two remaining claims clear. □

Lemma 4 shows that non-trivial, extremal meet-subsemilattices are always cover-preserving and topped. These two properties will be useful to assure that a doubling $L[K]$ is a meet-distributive lattice.

Lemma 4. *Let L be an (n,k)-extremal lattice with $k \geq 3$ and suppose that K is an $(n,k-1)$-extremal subsemilattice of L. Then, K is cover-preserving and topped. If $k \geq 4$, then K and L are atomistic with $A(K) = A(L)$.*

Proof. In case that $k = 3$ then K is B(2)-free, that is, K is a chain. By Lemma 3, K must be a maximal chain in order to have $\sum_{i=0}^{1} \binom{n}{i} = n+1$ elements. Hence, $1_K = 1_L$. The maximality of K guarantees that K is cover-preserving. Now, suppose that $k \geq 4$. Again by Lemma 3, we have that both K and L are atomistic. Since K has n atoms, 0_K must be covered by n elements in L. But this is possible only if $0_L = 0_K$, because L also has n atoms. This forces $A(K) = A(L)$ as well as $1_L \in K$, because 1_L is the only element that upper bounds each $a \in A(K)$. To prove that K is cover-preserving, we apply Lemma 3 twice, obtaining $|A_y \setminus A_x| = 1$ for every $x, y \in K$ with $x \prec_K y$ as well as $|A_y \setminus A_x| = 1$ for every $x, y \in L$ with $x \prec_L y$. Both conditions hold simultaneously only if the implication $x \prec_K y \Rightarrow x \prec_L y$ holds, i.e., if K is cover-preserving. □

A *complete meet-embedding* is a meet-embedding which preserves arbitrary meets, including $\bigwedge \emptyset$. As a consequence, the greatest element of one lattice gets mapped to the greatest element of the other. Images of complete meet-embeddings are topped meet-subsemilattices. This notion is required for the following simple fact, which aids us in the construction of sequences of (n,k)-extremal lattices with fixed n and growing k. In Proposition 9, the symbol $K[J]$ (for instance) means actually the doubling of the image of J under the corresponding embedding.

Proposition 9. *Suppose that J, K and L are lattices with complete meet-embeddings $\mathcal{E}_1 : J \to K$ and $\mathcal{E}_2 : K \to L$. Then, there exists a complete meet-embedding from $K[J]$ into $L[K]$.*

Proof. The fact that $K[J]$ and $L[K]$ are lattices comes from Proposition 5. Of course, there is an induced embedding from $\dot{J}$ into $\dot{K}$, but for which we will use the same symbol $\mathcal{E}_1$. The mapping $\mathcal{E}_3 : K[J] \to L[K]$ defined by $\mathcal{E}_3(x) = \mathcal{E}_1(x)$ for $x \in \dot{J}$ and $\mathcal{E}_3(x) = \mathcal{E}_2(x)$ for $x \in K$ may be checked as being a complete meet-embedding. □

As mentioned after Proposition 7, we will make use of an operation which doubles an extremal meet-subsemilattice of an extremal lattice. The next theorem shows that the lattice produced by this operation is indeed extremal.

Theorem 3. *Let L be an $(n-1,k)$-extremal lattice with $n \geq 2$ and $k \geq 3$ and suppose that K is an $(n-1,k-1)$-extremal meet-subsemilattice of L. Then, $L[K]$ is an (n,k)-extremal lattice.*

Proof. Lemma 3 guarantees that L is atomistic and meet-distributive. Moreover, Lemma 4 guarantees that K is cover-preserving and topped, so that, in particular, $L[K]$ is a meet-distributive lattice, as a consequence of Proposition 6. To prove that $L[K]$ is (n,k)-extremal it is sufficient to show that $L[K]$ is B(k)-free,

because of Proposition 7. We will do so by proving that every element of $L[K]$ has a representation through join-irreducibles of size at most $k-1$. This indeed suffices because $L[K]$ is meet-distributive, so that Lemma 2 may be applied. Observe that $J(L[K]) = A(L) \cup \dot{0_K}$, since L is atomistic.

Suppose that $k = 3$. In this case K is a chain, and a maximal one because of Lemma 4. Let $x \in L[K]$. If $x \in L$, then Lemma 2 implies that $x = \bigvee S$ for some $S \subseteq A(L), |S| \leq 2$ and the same representation may be used in $L[K]$. If $x \notin L$, then $x = \dot{y} = y \vee \dot{0_K}$ for some $y \in K$. If $y \in A(L)$, we are done. Otherwise, take $z, w \in A(L)$ such that $z \vee w = y$ and thus $x = z \vee w \vee \dot{0_K}$. Exactly one among z and w belong to K. Without loss of generality, let it be z. Then, it is clear that $\dot{0_K} \prec z \vee \dot{0_K}$ and that $z \vee \dot{0_K} < w \vee \dot{0_K}$, since there exists only one element covering $\dot{0_K}$. Hence, $x = z \vee w \vee \dot{0_K} = w \vee \dot{0_K}$.

Suppose that $k \geq 4$. As noted after Proposition 5 one has that K is, by itself, a lattice. Moreover, Lemma 4 guarantees that K is atomistic with $A(L) = A(K)$. Let $x \in L[K]$. If $x \in L$ then, in L, $x = \bigvee S$ for some $S \subseteq A(L) \subseteq J(L[K])$ with $|S| \leq k-1$, because of Lemma 2 and the fact that L is B(k)-free. Of course, S is also a representation of x in $L[K]$. If $x \notin L$, then $x = \dot{y}$ for some $\dot{y} \in \dot{K}$. Since K is B($k-1$)-free, it follows that, in K, $y = \bigvee S$ for some $S \subseteq A(K) \subseteq J(L[K])$ with $|S| \leq k-2$, once again as a consequence of Lemma 2. Clearly, in $L[K]$, one has $\dot{y} = \bigvee \dot{S} = \dot{0_K} \vee \bigvee S$, where the last equality follows from the fact that $\dot{z} = z \vee \dot{0_K}$ for every $z \in S$. Thus, we have a representation of $\dot{y} = x$ through no more than $k-1$ join-irreducible elements of $L[K]$. □

Corollary 1 describes how (n,k)-extremal lattices can be non-deterministically constructed. In particular, the upper bound present in Theorem 2 is the best possible.

Corollary 1. *For every n and k, there exists at least one (n,k)-extremal lattice.*

Proof. Define a partial function Φ satisfying

$$\Phi : \mathbb{N}^* \times \mathbb{N}^* \to \mathcal{L}$$

$$(n,k) \mapsto \begin{cases} ([n], \subseteq), & \text{if } k = 1. \\ (\{\emptyset, \{1\}\}, \subseteq), & \text{if } k \geq 2, n = 1. \\ \Phi(n-1,k)[\mathcal{E}(\Phi(n-1,k-1))], & \text{if } n,k \geq 2 \text{ and there exists a complete meet-embedding } \mathcal{E} : \Phi(n-1,k-1) \to \Phi(n-1,k). \end{cases},$$

where $\mathcal{L}$ is the class of all lattices. We prove by induction on n that $\Phi(n,k)$ is a total function. The cases $n = 1$ and $n = 2$ are trivial. Let $n \in \mathbb{N}$ with $n \geq 3$ and suppose that $\Phi(n-1,k)$ is defined for every $k \in \mathbb{N}^*$. Let $k \in \mathbb{N}, k \geq 2$. By the induction hypothesis, the values $\Phi(n-1,k)$ and $\Phi(n-1,k-1)$ are defined. If $k = 2$, then $\Phi(n-1,k-1)$ is a trivial lattice and the existence of a complete meet-embedding into $\Phi(n-1,k)$ is clear and, thereby, $\Phi(n,k)$ is defined. We therefore assume $k \geq 3$. By the definition of Φ, one has that $\Phi(n-1,k) = \Phi(n-$

$2,k)[\mathcal{E}(\Phi(n-2,k-1))]$ and that $\Phi(n-1,k-1) = \Phi(n-2,k-1)[\mathcal{F}(\Phi(n-2,k-2))]$ for some pair of complete meet-embeddings $\mathcal{E}$ and $\mathcal{F}$. Applying Proposition 9 with $\Phi(n-2,k-2), \Phi(n-2,k-1)$ and $\Phi(n-2,k)$ results in the existence of a complete meet-embedding $\mathcal{G} : \Phi(n-1,k-1) \to \Phi(n-1,k)$, which yields that $\Phi(n,k)$ is defined. Since k is arbitrary, every $\Phi(n,k)$ is defined. The (n,k)-extremality of each lattice can be proved by induction on n as well and by invoking Theorem 3. □

Figure 3 depicts the diagrams of nine (n,k)-extremal lattices which are constructible by Corollary 1. It is true that, in general, (n,k)-extremal lattices are not unique up to isomorphism: note that the $(3,3)$ and $(4,3)$-extremal lattices in Figure 3 are also present in Figure 2 as the lattices $\mathcal{C}_2$ and $\mathcal{C}_3$. The lattice $\mathcal{C}_4$, depicted in that same figure, is a $(4,3)$-extremal lattice which is not isomorphic to $\mathcal{C}_3$. We shall, however, show in the next section that every extremal lattice arises from the construction described in Corollary 1.

6 Characterization of extremal lattices

In the last section, we constructed lattices whose sizes are exactly the upper bound present in Theorem 2. In this section, we will show that every lattice meeting those requirements must be obtained from our construction.

Lemma 5. *Let L be an atomistic lattice, a an atom and c a coatom with $A_c = A(L) \setminus \{a\}$. Then, the mapping $x \overset{\mathcal{E}}{\mapsto} c \wedge x$ is a complete meet-embedding of $\uparrow a$ into $\downarrow c$ such that $\mathcal{E}(x) \prec x$ for every $x \in \uparrow a$.*

Proof. The fact that $\mathcal{E}$ preserves non-empty meets is clear, since c is a fixed element. Also, 1_L is mapped to $c = 1_{\downarrow c}$, so that $\mathcal{E}$ preserves arbitrary meets. Note that

$$A_{\mathcal{E}(x)} = A_{c \wedge x} = A_x \cap A_c = A_x \setminus \{a\}.$$

Hence, $\mathcal{E}(x) \prec x$ as well as $\mathcal{E}(x) \vee a = x$. The latter implies injectivity. □

The next theorem shows that every extremal lattice is constructible by the process described in Corollary 1, and can be seen as a converse of that result.

Theorem 4. *Let L be an (n,k)-extremal lattice with $k \geq 3$. Then, $L = J \dot{\cup} K$ where J is an $(n-1,k)$-extremal lattice and K is an $(n-1,k-1)$-extremal lattice. Moreover, there exists a complete meet-embedding $\mathcal{E} : K \to J$ such that $\mathcal{E}(x) \prec x$ for every $x \in K$. In particular, $L \cong J[\mathcal{E}(K)]$.*

Proof. From Lemma 3, one has that L is atomistic and we may take an atom a which is an extremal point of 1_L, that is, $A(L) \setminus \{a\} = A_c$ with c being a coatom of L. Consider the lattices $J = \downarrow c$ and $K = \uparrow a$. Observe that $L = J \dot{\cup} K$ and let $\mathcal{E} : K \to J$ be a complete meet-embedding provided by Lemma 5. Clearly, J has $n-1$ atoms and is $\mathrm{B}(k)$-free, therefore, $|J| \leq f(n-1,k)$. Moreover, K must be $\mathrm{B}(k-1)$-free: indeed, if there existed $B \cong \mathrm{B}(k-1)$ inside K, then

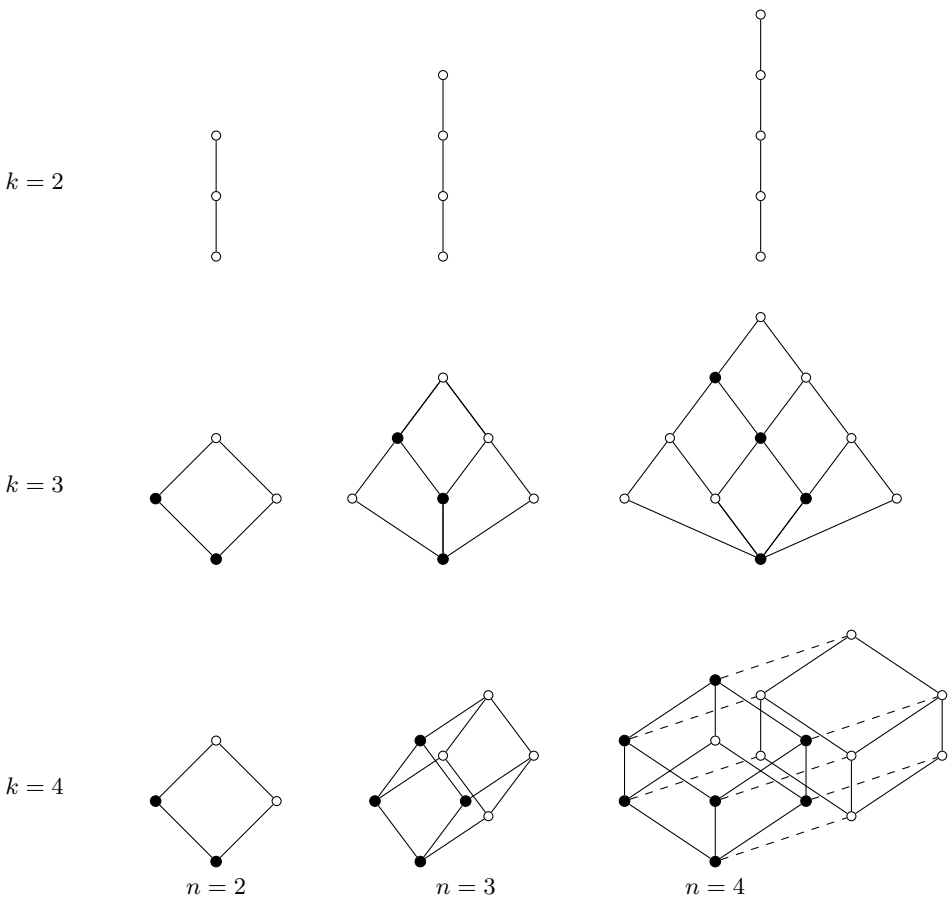


Fig. 3: Diagrams of (n, k)-extremal lattices with $2 \leq n, k \leq 4$. Elements shaded in black represent the doubled $(n-1, k-1)$-extremal lattice.

$B \cup \mathcal{E}(B)$ would be a boolean lattice with k atoms inside J, which is impossible. The lattice K has at most $n-1$ atoms, and consequently $|K| \leq f(n-1, k-1)$, since the function $n \mapsto \sum_{i=0}^{k-1} \binom{n}{i}$ is monotonic increasing. Now, we have that $|J| + |K| = |L| = f(n, k) = f(n-1, k) + f(n-1, k-1)$, where the last equality follows from Proposition 3. Since $|J| \leq f(n-1, k)$ and $|K| \leq f(n-1, k-1)$, those two inequalities must hold with equality. Therefore, J and K are, respectively, $(n-1, k)$ and $(n-1, k-1)$-extremal. □

7 Conclusion and related work

We showed an upper bound for the number of minimal generators of a context which is sharp. Extremal lattices were constructed and also characterized. The rôle played by contranominal scales in formal contexts may be seen as analogous

as that of cliques in simple graphs, when one considers extremality with respect to the number of edges. The Sauer-Shelah Lemma [8] provides an upper bound which is similar to that of Theorem 2. This is not a coincidence, because it can be shown, not without some effort, that the condition of a concept lattice being $\mathrm{B}(k)$-free is equivalent to the family of its extents not shattering a set of size k. As for the sharpness of the bound (which we prove in Section 5), in our case it is non-trivial, whereas the sharpness for the result of Sauer and Shelah is immediate.

8 Acknowledgements

We want to deeply thank Bernhard Ganter for the invaluable feedback and fruitful discussions.

References

1. Alexandre Albano. Upper bound for the number of concepts of contranominal-scale free contexts. In *Formal Concept Analysis - 12th International Conference, ICFCA 2014, Cluj-Napoca, Romania, June 10-13, 2014. Proceedings*, pages 44–53, 2014.
2. Alexandre Albano and Alair Pereira do Lago. A convexity upper bound for the number of maximal bicliques of a bipartite graph. *Discrete Applied Mathematics*, 165(0):12 – 24, 2014. 10th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2011).
3. David Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information Processing Letters*, 51(4):207–211, 1994.
4. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin-Heidelberg, 1999.
5. Sergei O. Kuznetsov. On computing the size of a lattice and related decision problems. *Order*, 18(4):313–321, 2001.
6. Erich Prisner. Bicliques in graphs I: bounds on their number. *Combinatorica*, 20(1):109–117, 2000.
7. Dieter Schütt. Abschätzungen für die Anzahl der Begriffe von Kontexten. Master's thesis, TH Darmstadt, 1987.
8. Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41(1):247–261, 1972.
9. Thomas Worsch. Lower and upper bounds for (sums of) binomial coefficients, 1994.

An Aho-Corasick Based Assessment of Algorithms Generating Failure Deterministic Finite Automata

Madoda Nxumalo[1], Derrick G. Kourie[2,3], Loek Cleophas[2,4], and Bruce W. Watson[2,3]

[1] Computer Science, Pretoria University, South Africa
[2] FASTAR Research, Information Science, Stellenbosch University, South Africa
[3] Centre for Artificial Intelligence Research, CSIR Meraka Institute, South Africa
[4] Foundations of Language Processing, Computer Science, Umeå University, Sweden
{madoda,derrick,loek,bruce}@fastar.org — http://www.fastar.org

Abstract. The Aho-Corasick algorithm derives a failure deterministic finite automaton for finding matches of a finite set of keywords in a text. It has the minimum number of transitions needed for this task. The DFA-Homomorphic Algorithm (DHA) algorithm is more general, deriving from an arbitrary complete deterministic finite automaton a language-equivalent failure deterministic finite automaton. DHA takes formal concepts of a lattice as input. This lattice is built from a state/out-transition formal context that is derived from the complete deterministic finite automaton. In this paper, three general variants of the abstract DHA are benchmarked against the specialised Aho-Corasick algorithm. It is shown that when heuristics for these variants are suitably chosen, the minimality attained by the Aho-Corasick algorithm can be closely approximated. A published non-lattice-based algorithm is also shown to perform well in experiments.

Keywords: Failure deterministic finite automaton, Aho-Corasick algorithm

1 Introduction

A deterministic finite automaton (DFA) defines a set of strings, called its language. It is represented as a graph with symbol-labelled transitions between states. There are efficient algorithms to determine whether an input string belongs to the DFA's language. An approach to reducing DFA memory requirements is the use of so-called failure deterministic finite automata (FDFAs, also defined in Section 2). An FDFA can be used to define the same language as a DFA with a reduced number of transitions and hence, reduced space required to store transition information. Essentially, this is achieved by replacing certain DFA state transitions by so-called failure transitions. A small additional computational cost is incurred in recognising whether given strings are part of the

language. By using a trie (the term used for a DFA graph that is a tree) and failure transitions, Aho and Corasick [1] generalised the so-called KMP algorithm [7] to multi-keyword pattern matching. There are two versions of their algorithm: the so-called optimal one, which we call *aco*, and a failure one, *acf*. *aco* builds a minimal DFA to find all matches of a given keyword set in a text. *acf* builds, in a first step, a trie using all prefixes of words from the set. Each state therefore represents a string which is the prefix of a keyword. Moreover, the string of a state is spelled out by transitions that connect the start state of the trie to that state. In a second step, *aco* then inserts a failure transition from each state of the trie to some other state. To briefly illustrate the nature failure transitions, suppose p is a state in the trie representing the string and keyword she and suppose q is another state representing the prefix string he of the keyword $hers$. Then a transition from p to q would indicate that he is the longest suffix of she that matches a prefix of some other keyword. With appropriate further elaboration (details may be found in [1, 11]) the output of *acf* is an FDFA that is language equivalent to the *aco* one It can also be shown that *acf* is minimal in the following sense. No other FDFA that is language-equivalent to *aco* can have fewer transitions than *acf*

An algorithm proposed in [8], called the DFA-Homomorphic Algorithm (DHA), constructs from *any* complete DFA a language-equivalent FDFA. As predicted by the theory [2], the resulting FDFA is not necessarily minimal. The abstract version of the algorithm described in [8] involves the construction of a concept lattice as explained in Section 2. The original version of the algorithm leaves a number of decisions as nondeterministic choices. It also strives for minimality by following a "greedy" heuristic in respect of information embedded in the lattice. However, concrete versions of DHA and the effect of this heuristic have not been tested. Here we propose various alternative concrete versions of the algorithm for different deterministic choices.The *acf* FDFAs provide a benchmark for assessing the performance of these concrete variants of DHA. An *aco* DFA is used as input to each of several variants of the DHA and the resulting DHA-FDFAs are compared against the *acf* version.

An alternative approach to constructing FDFAs from an arbitrary DFA has been proposed by Kumar et al [10][5]. Their technique is based on finding the maximal spanning tree [9] of a suitably weighted nondirected graph that reflects the structure of the underlying DFA. Two algorithms were proposed. One is based on a maximal spanning tree, and the other on a redefined maximal spanning tree. Further details about their algorithms may be found in their original publication. The original maximal spanning tree based algorithm was included in our comparative study for its performance assessment using *acf* as the benchmark.

[5] Their research uses different terminology to that given above. They refer to an FDFA as a delayed-input DFA (abbreviated to D^2FA) and failure transitions are called default transitions.

Various other FDFA related research has been conducted for certain limited contexts. See [5] for an overview. A recent example is discussed in [4], where ideas from [8] were used to modify construction of a so-called factor oracle automaton to use failure transitions, saving up to 9% of symbol transitions.

Section 2 provides the formal preliminaries relevant to this research. In Section 3 we introduce the deterministic variants of the DHA that are subsequently benchmarked. Section 4 outlines the investigation's experimental environment, the data generated, the methods of assessment and the results. Section 5 draws conclusions and points to further research work currently underway.

2 Preliminaries

An *alphabet* is a set of symbols, Σ, of size $|\Sigma|$, and Σ^* denotes the set of all sequences over this alphabet, including the empty sequence, denoted by ϵ. A *string* (or word), s, is an element of Σ^* and its length is denoted $|s|$. Note that $|\epsilon| = 0$. The *concatenation* of strings p and q is represented as pq. If $s = pqw$ then q is a *substring* of s, p and pq are *prefixes* of s and q and qw are *suffixes* of s. Moreover, q is a *proper* substring iff $\neg((p = \epsilon) \vee (w = \epsilon))$. Similarly, pq is a proper prefix iff $w \neq \epsilon$ and qw is a proper suffix iff $p \neq \epsilon$.

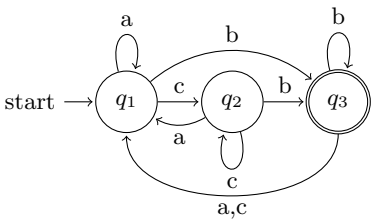
A deterministic finite automata (DFA) is a quintuple, $D = (Q, \Sigma, \delta, F, q_s)$, where Q is a finite set of states; Σ is an alphabet; $\delta \in Q \times \Sigma \nrightarrow Q$ is the (possibly partial) symbol transition function mapping symbol/state pairs to states; $q_s \in Q$ is the start state; and $F \subseteq Q$ is a set of final states. If δ is a total function, then the DFA is called *complete*. In the case of a complete DFA, the *extension of* δ is defined as $\delta^* \in Q \times \Sigma^* \longrightarrow Q$ where $\delta^*(p, \epsilon) = p$ and if $\delta(p, a) = q$ and $w \in \Sigma^*$, then $\delta^*(p, aw) = \delta^*(q, w)$. A finite string, w, is said to be *accepted* by the DFA iff $\delta^*(q_s, w) \in F$. The language of a DFA is the set of accepted strings.

A failure DFA (FDFA) is a six-tuple, $(Q, \Sigma, \delta, \mathfrak{f}, F, q_s)$, where $D = (Q, \Sigma, \delta, F, q_s)$ is a (not necessarily complete) DFA and $\mathfrak{f} \in Q \nrightarrow Q$ is a (possibly partial) failure transition function. For all $a \in \Sigma$ and $p \in Q$, the functions δ and $\mathfrak{f}$ are related in the following way: $\mathfrak{f}(p) = q$ for some $q \in Q$ if $\delta(p, a)$ is not defined. The extension of δ in an FDFA context is similar to its DFA version in that $\delta^* \in Q \times \Sigma^* \longrightarrow Q$ and $\delta^*(p, \epsilon) = p$. However:

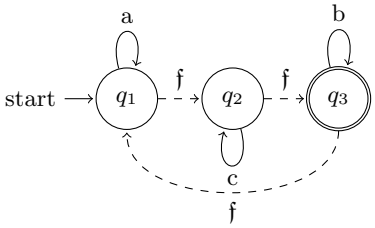
$$\delta^*(p, aw) = \begin{cases} \delta^*(q, w) \text{ if } \delta(p, a) = q \\ \delta^*(q, aw) \text{ if } \delta(p, a) \text{ is not defined and } \mathfrak{f}(p) = q \end{cases}$$

An FDFA is said to accept string $w \in \Sigma^*$ iff $\delta^*(q_s, w) \in F$. An FDFA's language is its set of accepted strings. It can be shown that every complete DFA has a language-equivalent FDFA and *vice-versa*. When constructing an FDFA from a DFA, care must be taken to avoid so-called divergent cycles of failure transitions because they lead to an infinite sequence of failure traversals in string processing algorithms. (Details are provided in [8].)

Subfigure 1a depicts a complete DFA for which $Q = \{q_1, q_2, q_3\}$ and $\Sigma = \{a, b, c\}$. Its start state is q_1 and q_3 is the only final state. Its symbol transitions are depicted as solid arrows between states. Subfigure 1b shows a language-equivalent FDFA where dashed arrows indicate the failure transitions. Note, for example that ab is in the language of both automata. In the DFA case, $\delta^*(q_1, ab) = \delta^*(q_1, b) = \delta^*(q_3, \epsilon)$ In the FDFA case, $\delta^*(q_1, ab) = \delta^*(q_1, b) = \delta^*(q_2, b) = \delta^*(q_3, b) = \delta^*(q_3, \epsilon)$.



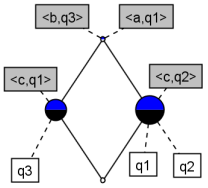
(a) $D = (Q, \Sigma, \delta, q_1, \{q_3\})$



(b) $F = (Q, \Sigma, \delta, \mathfrak{f}, q_1, \{q_3\})$

	$\langle a, q_1 \rangle$	$\langle b, q_3 \rangle$	$\langle c, q_2 \rangle$	$\langle c, q_1 \rangle$
q_1	X	X	X	
q_2	X	X	X	
q_3	X	X		X

(c) State/out-transition context



(d) State/out-transition lattice

Fig. 1: Example automata and state/out-transition lattice

This text relies on standard formal concept analysis terminology and definitions. (See, for example [6]). A so-called *state/out-transition* concept lattice can be derived from any DFA. The objects of its formal context are DFA states, $q \in Q$. Each attribute is a pair of the form $\langle a, p \rangle \in \Sigma \times Q$. A state q is deemed to have this attribute if $\delta(q, a) = p$, i.e. if q is the source of a transition on a to p. Subfigure 1c is the state/out-transition context for the DFA in Subfigure 1a and Subfigure 1d is the line diagram of the associated state/out-transition concept lattice. The latter subfigure shows two intermediate concepts, each larger than the bottom concept and smaller than the top, but not commensurate with one another. The right-hand side intermediate concept depicts the fact that states q_1 and q_2 (its extent) are similar in that each as a transition on symbol a to q_1, b to q_3 and on c to q_2 — i.e. the concept's intent is $\{\langle a, q_1 \rangle, \langle b, q_3 \rangle, \langle c, _{q2} \rangle\}$

Each concept in a state/out-transition lattice can be characterised by a certain value, called its *arc redundancy*. For a concept c it is defined as $ar(c) = (|int(c)| - 1) \times (|ext(c)| - 1)$, where $ext(c)$ and $int(c)$ denote the extent and intent of concept

c respectively. The arc redundancy of a concept represents the number of arcs that may be saved by doing the following:

1. singling out one of the states in the concept's extent;
2. at all the remaining states in the concept's extent, removing all out-transitions mentioned in the concept's intent;
3. inserting a failure arc from each of the states in step 2 to the singled out state in step 1.

The expression, $|ext(c)| - 1$ represents the number of states in step 2 above. At each such state, $|int(c)|$ symbol transitions are removed and a failure arc is inserted. Thus, $|int(c)| - 1$ is the total number of transitions saved at each of $|ext(c)| - 1$ states so that $ar(c)$ is indeed the total number of arcs saved by the above transformation.

The *positive arc redundancy* (PAR) set consists of all concepts whose arc redundancy is greater than zero.

3 The DHA Variants

For the DFA-Homomorphic Algorithm (DHA) to convert a DFA into a language equivalent FDFA, a three stage transformation of the DFA is undertaken. Initially, the DFA is represented as a state/out-transition context. From the derived concept lattices, the PAR set is extracted to serve as input for the DHA.

The basic DHA proposed in [8] is outlined in Algorithm 1. The variable O is used to keep track of states that are not the source of any failure transitions. This is to ensure that a state is never the source of more than one failure transition. Initially all states qualify. A concept c is selected and removed from PAR set, so that c is no longer available in subsequent iterations. The initial version of DHA proposed specifically selecting a concept, c, with maximum arc redundancy. The specification given here leaves open how the choice will be made.

From c's extent, one of the states, t, is chosen to be a failure transition target state. DHA gives no specific criteria for which state in $ext(c)$ to choose. The remaining set of states in $ext(c)$ is denoted by $ext'(c)$. Then, for each state s in $ext'(c)$ that qualifies to be the source of a failure transition (i.e. that is also in O) all transitions in $int(c)$ are removed from s and a failure transition is installed from s to t. Because state s has become a failure transition source state whose target state is t, it may no longer be the source of any other failure transition, and so is removed from O. These steps are repeated until it is no longer possible to install any more failure transitions. It should be noted that in this particular formulation of the abstract algorithm the PAR set is not recomputed to reflect changes in arc redundancy as the DFA is progressively transformed into an FDFA. This does not affect the correctness of the algorithm, but may affect its optimality. Investigating such effects is not within the scope of this study. The third and fifth lines of Algorithm 1, namely

$c := selectAConcept(PAR)$ and
$t := getAnyState(ext(c))$ respectively.

are non-specific in the original formulation of DHA.

Three variants of the algorithm are proposed with respect to the third line. For convenience we represent each variant by a conjunct on the right hand side of an assignment where c is the assignment target. This, of course, is a slight abuse of notation since c is not the outcome of a logical operation, but the selection of a concept from the PAR set according to a criterion represented by $h\_mar(PAR)$, $h\_me(PAR)$ or $h\_mi(PAR)$. Each selection option a different greedy heuristic for choosing concept c from the PAR set. By greedy we mean that an element from the set is selected, based on some maximal or minimal feature, without regard to possible opportunities lost in the forthcoming iterations by making these selections. In addition to these heuristics, a single heuristic is proposed for the fifth line relating to choosing the target state, t, for the failure transitions. These choices are illustrated as colour-coded assignment statements shown in the skeleton Algorithm 2 and are now briefly explained. The rationale for these heuristics will be discussed a section below.

Algorithm 1

$O := Q$; $PAR := \{c \mid ar(c) > 0\}$;
do $((O \neq \emptyset) \wedge (PAR \neq \emptyset)) \rightarrow$
  $c := SelectConcept(PAR)$;
  $PAR := PAR \backslash \{c\}$;
  $t := getAnyState(ext(c))$;
  $ext'(c) := ext(c) \backslash \{t\}$;
  for each $(s \in ext'(c) \cap O) \rightarrow$
    if *a failure cycle is not created* $\rightarrow$
      for each $((a, r) \in int(c)) \rightarrow$
        $\delta := \delta \setminus \{\langle s, a, r\rangle\}$
      rof;
      $\mathfrak{f}(s) := t$;
      $O := O \backslash \{s\}$;
    fi
  rof
od

Algorithm 2

$O := Q$; $PAR := \{c \mid ar(c) > 0\}$;
do $((O \neq \emptyset) \wedge (PAR \neq \emptyset)) \rightarrow$
  $c := h\_mar(PAR) \vee h\_me(PAR) \vee h\_mi(PAR)$
  $PAR := PAR \backslash \{c\}$
  $t := ClosestToRoot(c)$
  $ext'(c) := ext(c) \backslash \{t\}$;
  for each $(s \in ext'(c) \cap O) \rightarrow$
    if *a failure cycle is not created* $\rightarrow$
      for each $((a, r) \in int(c)) \rightarrow$
        $\delta := \delta \setminus \{\langle s, a, r\rangle\}$
      rof;
      $\mathfrak{f}(s) := t$;
      $O := O \backslash \{s\}$;
    fi
  rof
od

The heuristics for choosing concept c from the PAR set in each iteration are as follows: The h_mar heuristic: c is a PAR concept with a *maximum arc redundancy*. The h_mi heuristic: c is a PAR concept with a *maximum intent* size. The h_me heuristic: c is a PAR concept with a *minimum extent* size. Once one of these heuristics has been applied, the so-called *ClosestToRoot* heuristic is used to select a state t in $ext(c)$ to become the target state of failure transitions from each of the remaining states in $ext(c)$. The heuristic means that t is selected as the state in $ext(c)$ that is closest[6] to *aco*'s start state. Transition modifications are subsequently made on the FDFA produced to date, provided that a divergent failure cycle is not produced.

[6] Since a trie has no cycles, the notion of "closest" here simply means a state with the shortest path from the start state to that state.

4 The Experiment

The experiments were conducted on an Intel i5 dual core CPU machine, running Linux Ubuntu 14.4. Code was written in C++ and compiled under the GCC version 4.8.2 compiler.

It can easily be demonstrated that if there are no overlaps between proper prefixes and proper suffixes of keywords in a keyword set, then the associated *acf* FDFA's failure transitions will all loop back to its start state, and out *ClosestToRoot* heuristic will behave similarly. To avoid keyword sets that lead to such trivial *acf* FDFAs, the following keyword set construction algorithm was devised.

Keywords (also referred to as patterns) are from an alphabet set of size 10. Their lengths range from 5 to 60 characters. Keyword sets of sizes $5, 10, 15, \ldots, 100$ respectively are generated. For each of these 20 different set sizes, twelve alternative keyword sets are generated. Thus in total $12 \times 20 = 240$ keyword sets are available.

To construct a keyword set of size N, an initial N random strings are generated[7]. Each such string has random content taken from the alphabet and random length in the range 5 and 30. However, for reasons given below, only a limited number of these N strings, say M, are directly inserted into the keyword set. The set is then incrementally grown to the desired size, N, by repeating the following:

> Select a prefix of random length, say p, from a randomly selected string in the current keyword set. Remove a string, say w, from the set of strings not yet in the keyword set. Insert either pw or wp into the keyword set.

Steps are taken to ensure that there is a reasonable representation of each of these three differently constructed keywords in a given keyword set.

These keyword sets served as input to the SPARE-PARTS toolkit [12] to create the associated *acf* FDFAs and the *aco* DFAs. A routine was written to extract state/out-transition contexts from the *aco* DFAs. These contexts were used by the lattice construction software package known as FCART (version 0.9)[3] supplied to us by National Research University Higher School of Economics (Moscow, Russia). The DHA variants under test used resulting concept lattices to generate the FDFAs. As previously mentioned, a Kumar et al [10] algorithm was also implemented to generate FDFAs from the DFAs. These will be referenced as *kum* FDFAs.

Figures 2 and 3 give several views of the extent to which the DHA-based and *kum* FDFAs correspond with *acf* FDFAs. The experimental data is available online[8]. For notational convenience f^i_{jk} denotes the set of failure transitions of

[7] Note that all random selections mentioned use a pseudo-random number generator.
[8] The experimental data files can be found at this URL:
http : *//www.fastar.org/wiki/index.php?title* = *Conference_Papers#2015*.

the FDFA corresponding to k^{th} keyword set of size $5j$ that was generated by the algorithm variant $i \in FA \backslash \{aco\}$, where $k \in [1, 12]$, $j \in [1, 20]$ and $FA = \{acf, aco, mar, mi, me, kum\}$. Similarly, δ^i_{jk} refers to symbol transition sets of the associated FDFAs and, in this case, also the *aco* DFAs if $i = aco$. A dot notation in the subscript is used for averages. Thus, for some $i \in FA \backslash \{aco\}$ we use $|\mathfrak{f}^i_{j.}|$ to denote the average number of failure transitions in the i-type FDFAs produced by the 12 keyword sets of size $5j$, and similarly $|\delta^i_{j.}|$ represents the average number of symbol transitions.

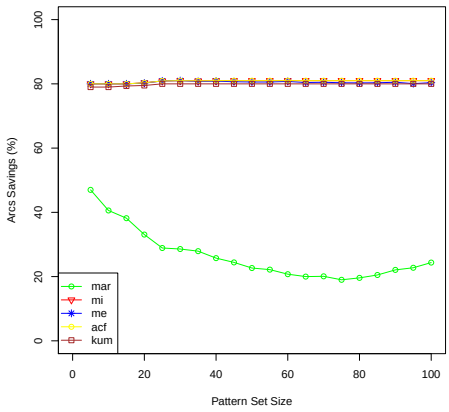
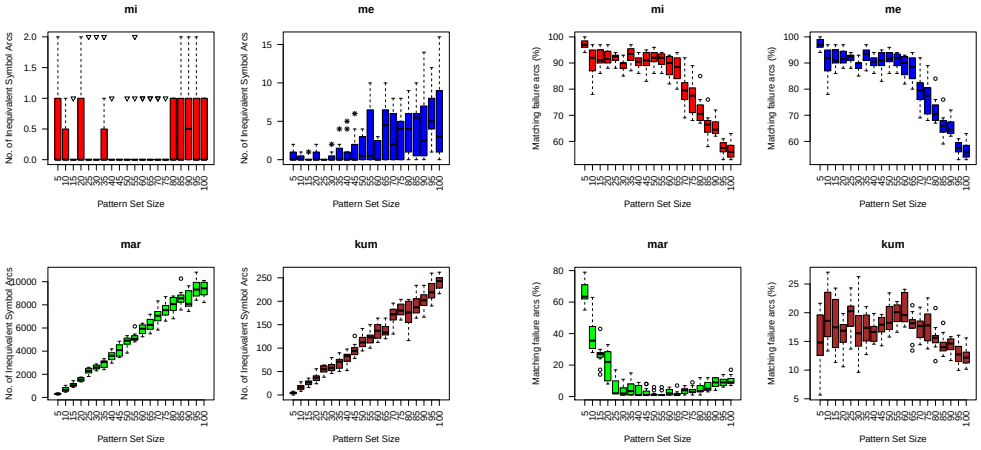


Fig. 2: $\dfrac{|\delta^{aco}_{j.}| - (|\delta^i_{j.}| + |\mathfrak{f}^i_{j.}|)}{|\delta^{aco}_{j.}|} \times 100$

Figure 2 shows how many more transitions *aco* automata require (as a percentage of *aco*) compared to each of the FDFA variants. Note that data has been averaged over the 12 keyword set samples for each of the set sizes and note that the FDFA transitions include both symbol and failure transitions. The minimal *acf* FDFAs attain an average savings of about 80% over all sample sizes and the *mi*, *me* and *kum* FDFAs track this performance almost identically. Although not clearly visible in the graph, the *me* heuristic shows a slight degradation for larger set sizes, while the *kum* FDFAs consistently perform about 1% to 2% worse. By way of contrast, the *mar* heuristic barely achieves a 50% savings for small sample sizes, and drops below a 20% savings for a sample size of about 75, after which there is some evidence that it might improve slightly.

The fact that the percentage transition savings of the various FDFA variants closely correspond to that of *acf* does not mean that the positioning of the failure and symbol transitions should show a one-to-one matching. The extent to which the transitions precisely match one another is shown in Figure 3. These box-and-



(a) $|\delta^{acf}_{jk}| - |\delta^{acf}_{jk} \cap \delta^{i}_{jk}|$

(b) $\frac{|\mathfrak{f}^{i}_{jk} \cap \mathfrak{f}^{acf}_{jk}|}{|\mathfrak{f}^{acf}_{jk}|} \times 100$

Fig. 3: Transition Matches

whisker plots show explicitly the median, 25^{th} and 75^{th} percentiles as well as outliers of each of the 12 sample keyword sets of a given size. Subfigure 3a shows the number of symbol transitions in *acf* FDFAs that do not correspond with those in *mi*, *me*, *mar* and *kum* respectively. Subfigure 3b shows the percentage of *acf* failure transitions matching those of the FDFAs generated by *mi*, *me*, *mar* and *kum* respectively.

The symbol transitions for the *mi* heuristic are practically identical to those of *acf*, differing by at most two. Differences are not significantly related to sample size. Differences for *me* are somewhat larger, increasing slightly with larger sample size, though still relatively modest in relation to the overall number of FDFA transitions. (There are $|Q| - 1$ transitions in the underlying trie.) In the cases of *mar* and *kum*, the differences are approximately linearly dependent on the size of the keyword set, reaching over 9000 and 250 respectively for keywords sets of size 100.

The failure transition differences in regard to *mi* and *me* show a very similar pattern as keyword size increases. Only in isolated instances do they fully match those of *acf*, but the matching correspondence drops from a median of more than 95% in the case of the smallest keywords sets to a median of about 50% for the largest keyword sets. The median *kum* failure transition correspondence with *acf* is in a range of about 12-18% for all pattern set sizes. However, in the case of *mar*, the degree of correspondence is much worse: at best the median value is just over 60% for small keyword sets, dropping close to zero for medium

range keyword set sizes, and then increasing slightly to about 10% for the largest keyword sets.

Overall, Figures 2 and 3 reveal that there is a variety of ways in which failure transitions may be positioned in an FDFA, and that lead to very good—in many cases even optimal—transition savings. It is interesting to note that even in the *kum* FDFAs, the total number of transition savings is very close to optimal, despite relatively large differences in the positioning of the transitions. However, the figures also show that this flexibility in positioning failure transitions to achieve good arc savings eventually breaks down, as in the case of the *mar* FDFAs.

One of the reasons for differences between *acf* FDFAs and the others is that some implementations of the *acf* algorithm, including the SPARE-PARTS implementation, inserts a failure arc at *every* state (except the start state), even if there is an out-transition on every alphabet symbol from a state. Such a failure arc is of course redundant. Inspection of the data showed that some of the randomly generated keyword sets lead to such "useless" failure transitions, but they are so rare that they do not materially affect the overall observations.

The overall rankings of the output FDFAs of the various algorithms to *acf* could broadly be stated as $mi > me > kum > mar$. This ranking is with respect to closeness of transition placement to *acf*. Since the original focus of this study was to explore heuristics for the DHA algorithm, further comments about the *kum* algorithm are reserved for the next section.

The rationale for the *mar* heuristic is clear: it will cause the maximum savings in transitions in a given iteration. It was in fact the initial criterion proposed in [8]. It is therefore somewhat surprising that it did not perform very well in comparison to other heuristics. It would seem that, in the present context, it is too greedy—i.e. by selecting a concept whose extent contains the set of states that can effect maximal savings such that in one iteration, it deleteriously eliminates from consideration concepts whose extent contains some of those states in subsequent iterations. Note that, being based on the maximum of the product of extent and intent sizes, it will tend to select concepts in the middle of the concept lattice.

When early trials in our data showed up *mar*'s relatively poor performance, the *mi* and *me* heuristics were introduced to prioritise concepts in the top or bottom regions of the lattice. These latter two heuristics will maximize the number of symbol transitions to be removed *per state* when replacing them with failure transitions, in so far as concepts with large intents tend to have small extents and *vice-versa*. Although such a relationship is, of course, data-dependent, random data tends in that direction, as was confirmed by inspection of our data.

These two heuristics appear to be rather successful at attaining *acf*-like FDFAs. However, the *ClosestToRoot* heuristic has also played a part in this success. Note that the *acf* failure transitions are designed to record that a suffix of a state's

string is also a prefix of some other state's string. Thus, $f(q) = p$ means that a suffix of state q's string is also a prefix of state p's string. However, since there may be several suffixes of q's string and several states whose prefixes meet this criterion, the definition of f requires that the *longest possible* suffix of q's string should be used. This ensures that there is only one possible state, p, in the trie whose prefix corresponds to that suffix. Thus, on the one hand, *acf* directs a failure transition "backwards" towards a state whose depth is less than that of the current state. On the other hand, *acf* selects a failure transition's target state to be as *far* as possible from the start state, because the suffix (and therefore also the prefix) used must be maximal in length.

The *ClosestToRoot* heuristic approximates the *acf* action in that it also directs failure transitions backwards towards the start state. However, by selecting a failure transition's target state to be as *close* as possible from the start state, it seems to contradict *acf* actions. It is interesting to note in Subfigure 3b that both *mi* and *me* show a rapid and more or less linear decline in failure transition matchings with respect to *acf* when pattern set size reaches about 65. We conjecture that for smaller keyword sizes, the *ClosestToRoot* heuristic does not conflict significantly with *acf*'s actions because there are few failure target states from which to choose. When keyword set sizes become greater, there is likely to be more failure target states from which to choose, and consequently less correspondence between the failure transitions chosen according to differing criteria.This is but one of several matters that has been left for further study.

5 Conclusions and Future Agenda

Our ultimate purpose is to investigate heuristics for building FDFAs from *generalised* complete DFAs—a domain where optimal behaviour is known *a priori* to be computationally hard. The comparison against *acf* FDFAs outlined above is a firm but limited starting point. The next step is to construct complete DFAs from randomly generated FDFAs and examine the extent to which the heuristics tested out in this study can reconstruct the latter from the former. Because generalised DFAs can have cycles, the *ClosestToRoot* heuristic will be generalised by using Dijktra's algorithm for calculating the shortest distance from the start state to each DFA state. It remains to be seen whether *mar* will perform any better in the generalised context.

The relatively small alphabet size of 10 was dictated by unavoidable growth in the size of the associated concept lattices. Even though suitable strategies for trimming the lattice (for example by not generating concepts with arc redundancy less than 2) are being investigated, it is recognised that use of DHA will always be constrained by the potential for the associated lattice to grow exponentially. Nevertheless, from a theoretical perspective a lattice-based DHA approach to FDFA generation is attractive because it encapsulates the solution space in which a minimal FDFA might be found—i.e. each ordering of its concepts maps

to a possible language-equivalent FDFA that can be derived from DFA and at least one such ordering will be a minimal FDFA.

The *kum* FDFA generation approach is not as constrained by space limitations as the DHA approach and in the present experiments it has performed reasonably well. In the original publication, a somewhat more refined version is reported that attempts to avoid unnecessary chains of failure transitions. Future research should examine the minimising potential of this refined version using generalised DFAs as input and should explore more fully the relationship between these *kum*-based algorithms and the DHA algorithms.

References

1. A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, 1975.
2. H. Björklund, J. Björklund, and N. Zechner. Compression of finite-state automata through failure transitions. *Theor. Comput. Sci.*, 557:87–100, 2014.
3. A. Buzmakov and A. Neznanov. Practical computing with pattern structures in FCART environment. In *Proceedings of the International Workshop "What can FCA do for Artificial Intelligence?" (FCA4AI at IJCAI 2013), Beijing, China, August 5, 2013.*, pages 49–56, 2013.
4. L. Cleophas, D. G. Kourie, and B. W. Watson. Weak factor automata: Comparing (failure) oracles and storacles. In J. Holub and J. Žďárek, editors, *Proceedings of the Prague Stringology Conference 2013*, pages 176–190, Czech Technical University in Prague, Czech Republic, 2013.
5. M. Crochemore and C. Hancart. Automata for matching patterns. In S. A. Rozenberg G., editor, *Handbook of Formal Languages*, volume 2, Linear Modeling: Background and Application, pages 399–462. Springer-Verlag, 1997. incollection.
6. B. Ganter, G. Stumme, and R. Wille, editors. *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*. Springer, 2005.
7. D. E. Knuth, J. H. M. Jr., and V. R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.
8. D. G. Kourie, B. W. Watson, L. Cleophas, and F. Venter. Failure deterministic finite automata. In J. Holub and J. Žďárek, editors, *Proceedings of the Prague Stringology Conference 2012*, pages 28–41, Czech Technical University in Prague, Czech Republic, 2012.
9. J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.
10. S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. S. Turner. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In *Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Pisa, Italy, September 11-15, 2006*, pages 339–350, 2006.
11. B. W. Watson. *Taxonomies and Toolkits of Regular Language Algorithms*. PhD thesis, Eindhoven University of Technology, Sept. 1995.
12. B. W. Watson and L. G. Cleophas. SPARE Parts: a C++ toolkit for string pattern recognition. *Softw., Pract. Exper.*, 34(7):697–710, 2004.

Context-Aware Recommender System Based on Boolean Matrix Factorisation

Marat Akhmatnurov and Dmitry I. Ignatov

National Research University Higher School of Economics, Moscow
dignatov@hse.ru

Abstract. In this work we propose and study an approach for collaborative filtering, which is based on Boolean matrix factorisation and exploits additional (context) information about users and items. To avoid similarity loss in case of Boolean representation we use an adjusted type of projection of a target user to the obtained factor space. We have compared the proposed method with SVD-based approach on the MovieLens dataset. The experiments demonstrate that the proposed method has better MAE and Precision and comparable Recall and F-measure. We also report an increase of quality in the context information presence.

Keywords: Boolean Matrix Factorisation, Formal Concept Analysis, Recommender Algorithms, Context-Aware Recommendations

1 Introduction

Recommender Systems have recently become one of the most popular applications of Machine Learning and Data Mining. Their primary aim is to help users to find proper items like movies, books or goods within an underlying information system. Collaborative filtering recommender algorithms based on matrix factorisation (MF) techniques are now considered industry standard [1]. The main assumption here is that similar users prefer similar items and MF helps to find (latent) similarity in the reduced space efficiently.

Among the most often used types of MF we should definitely mention Singular Value Decomposition (SVD) [2] and its various modifications like Probabilistic Latent Semantic Analysis (PLSA) [3]. However, several existing factorisation techniques, for example, non-negative matrix factorisation (NMF) [4] and Boolean matrix factorisation (BMF) [5], seem to be less studied in the context of Recommender Systems. Another approach similar to MF is biclustering, which has also been successfully applied in recommender system domain [6,7]. For example, Formal Concept Analysis (FCA) [8] can be also used as a biclustering technique and there are several examples of its applications in the recommender systems domain [9,10]. A parameter-free approach that exploits a neighbourhood of the object concept for a particular user also proved its effectiveness [11]; it has a predecessor based on object-attribute biclusters [7] that also capture the neighbourhood of every user and item pair in an input formal context. Our

previous approach based on FCA exploits Boolean factorisation based on formal concepts and follows user-based k-nearest neighbours strategy [12].

The aim of this study is to continue comparing the recommendation quality of several aforementioned techniques on the real dataset and investigation of methods' interrelationship and applicability. In particular, in our previous study, it was especially interesting to conduct experiments and compare recommendation quality in case of a numeric input matrix and its scaled Boolean counterpart in terms of Mean Absolute Error (MAE) as well as Precision and Recall. Our previous results showed that the BMF-based approach is of comparable quality with the SVD-based one [12]. Thus, one of the next steps is definitely usage of auxiliary information containing users' and items' features, i.e. so called context information (for BMF vs SVD see section 4).

Another novelty of the paper is defined by the fact that we have adjusted the original Boolean projection of users to the factor space by support-based weights that results in a sufficient quality increase. We also investigate the approximate greedy algorithm proposed in [5] in the recommender setting, which tends to generate factors with large number of users, and more balanced (in terms of ratio between users' and items' number per factor) modification of the Close-by-One algorithm [13].

The practical significance of the paper is determined by the demand of recommender systems' industry, that is focused on gaining reliable quality in terms of average MAE.

The rest of the paper consists of five sections. Section 2 is an introductory review of the existing MF-based approaches to collaborative filtering. In section 3 we describe our recommender algorithm which is based on Boolean matrix factorisation using closed sets of users and items (that is FCA). Section 4 contains results of experimental comparison of two MF-based recommender algorithms by means of cross-validation in terms of MAE, Precision, Recall and F-measure. The last section concludes the paper.

2 Introductory review

In this section we briefly describe two approaches to the decomposition of real-valued and Boolean matrices. In addition we provide the reader with the general scheme of user-based recommendation that relies on MF and a simple way of direct incorporation of context information into MF-based algoritms.

2.1 Singular Value Decomposition

Singular Value Decomposition (SVD) is a decomposition of a rectangular matrix $A \in \mathbb{R}^{m \times n} (m > n)$ into a product of three matrices

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T, \tag{1}$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix such that $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$ and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. The columns of the matrix U and V are called singular vectors, and the numbers σ_i are singular values.

In the context of recommendation systems rows of U and V can be interpreted as vectors of user's and items's attitude to a certain topic (factor), and the corresponding singular values as importance of the topic among the others. The main disadvantages are the dense outputted decomposition matrices and negative values of factors which are difficult to interpret.

The advantage of SVD for recommendation systems is that this method allows to obtain a vector of user's attitude to certain topics for a new user without SVD decomposition of the whole matrix.

The computational complexity of SVD according to [2] is $O(mn^2)$ floating-point operations if $m \geq n$ or more precisely $2mn^2 + 2n^3$.

2.2 Boolean Matrix Factorisation based on FCA

Description of FCA-based BMF. Boolean matrix factorisation (BMF) is a decomposition of the original matrix $I \in \{0,1\}^{n \times m}$, where $I_{ij} \in \{0,1\}$, into a Boolean matrix product $P \circ Q$ of binary matrices $P \in \{0,1\}^{n \times k}$ and $Q \in \{0,1\}^{k \times m}$ for the smallest possible number of k. We define Boolean matrix product as follows:

$$(P \circ Q)_{ij} = \bigvee_{l=1}^{k} P_{il} \cdot Q_{lj}, \tag{2}$$

where $\bigvee$ denotes disjunction, and $\cdot$ conjunction.

Matrix I can be considered a matrix of binary relations between set X of objects (users), and a set Y of attributes (items that users have evaluated). We assume that xIy iff the user x evaluated object y. The triple (X, Y, I) clearly forms a formal context[1].

Consider a set $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, a subset of all formal concepts of context (X, Y, I), and introduce matrices $P_{\mathcal{F}}$ and $Q_{\mathcal{F}}$:

$$(P_{\mathcal{F}})_{il} = \begin{cases} 1, i \in A_l, \\ 0, i \notin A_l, \end{cases} \quad (Q_{\mathcal{F}})_{lj} = \begin{cases} 1, j \in B_l, \\ 0, j \notin B_l. \end{cases},$$

where (A_l, B_l) is a formal concept from F.

We can consider decomposition of the matrix I into binary matrix product $P_{\mathcal{F}}$ and $Q_{\mathcal{F}}$ as described above. The theorems on universality and optimality of formal concepts are proved in [5].

There are several algorithms for finding $P_{\mathcal{F}}$ and $Q_{\mathcal{F}}$ by calculating formal concepts based on these theorems [5]. The approximate algorithm we use for comparison (Algorithm 2 from [5]) avoids computation of all possible formal concepts and therefore works much faster [5]. Time estimation of the calculations in the worst case yields $O(k|G||M|^3)$, where k is the number of found factors, $|G|$ is the number of objects, $|M|$ is the number of attributes.

[1] We have to omit basic FCA definitions; for more details see [8].

2.3 Contextual information

Contextual Information is a multi-faceted notion that is present in several disciplines. In the recommender systems domain, the context is any auxiliary information concerning users (like gender, age, occupation, living place) and/or items (like genre of a movie, book or music), which shows not only a user's mark given to an item but explicitly or implicitly describes the circumstances of such evaluation (e.g., including time and place) [15].

From the representational viewpoint context[2] can be described by a binary relation, which shows that a user or an item possesses a certain attribute-value pair. In case the contextual information is described by finite-valued attributes, it can be represented by finite number of binary relations; otherwise, when we have countable or continuous values, their domains can be split into (semi)intervals (cf. scaling in FCA). As a result one may obtain a block matrix:

$$I = \begin{bmatrix} R & C_{user} \\ C_{item} & O \end{bmatrix},$$

where R is a utility matrix of users' ratings to items, C_{user} represents context information of users, C_{item} contains context iformation of items and O is zero-filled matrix.

Table 1. Adding auxialiry (context) information

	Movies						Gender		Age		
	Brave Heart	Termi-nator	Gladi-ator	Million-aire from ghetto	Hot Snow	God-father	M	F	0-20	21-45	46+
Anna	5		5	5		2		+	+		
Vladimir		5	5	3		5	+			+	
Katja	4		4	5		4		+		+	
Mikhail	3	5	5			5	+			+	
Nikolay			2		5	4	+				+
Olga	5	3	4	5				+	+		
Petr	5			4	5	4	+				+
Drama	+		+	+	+	+					
Action		+	+		+	+					
Comedy	+			+							

In case of more complex rating's scale the ratings can be reduced to binary scale (e.g., "*like/dislike*") by binary thresholding or by FCA-based scaling.

[2] In order to avoid confusion, please note that formal context is a different notion.

2.4 General scheme of user-based recommendations

Once a matrix of ratings is factorised we need to learn how to compute recommendations for users and to evaluate whether a particular method handles this task well.

For the factorised matrices already well-known algorithm based on the similarity of users can be applied, where for finding k nearest neighbors we use not the original matrix of ratings $R \in \mathbb{R}^{m \times n}$, but the matrix $I \in \mathbb{R}^{m \times f}$, where m is a number of users, and f is a number of factors. After the selection of k users, which are the most similar to a given user, based on the factors that are peculiar to them, it is possible, based on collaborative filtering formulas to calculate the prospective ratings for a given user.

After generation of recommendations the performance of the recommender system can be estimated by measures such as MAE, Precision and Recall.

Collaborative recommender systems try to predict the utility (in our case ratings) of items for a particular user based on the items previously rated by other users.

Memory-based algorithms make rating predictions based on the entire collection of previously rated items by the users. That is, the value of the unknown rating $r_{u,m}$ for a user u and item m is usually computed as an aggregate of the ratings of some other (usually, the k most similar) users for the same item m:

$$r_{u,m} = aggr_{\tilde{u} \in \tilde{U}} r_{\tilde{u},m},$$

where $\tilde{U}$ denotes a set of k users that are the most similar to user u, who have rated item m. For example, the function $aggr$ may be weighted average of ratings [15]:

$$r_{u,m} = \sum_{\tilde{u} \in \tilde{U}} sim(\tilde{u}, u) \cdot r_{\tilde{u},m} / \sum_{\tilde{u} \in \tilde{U}} sim(u, \tilde{u}). \tag{3}$$

The similarity measure between users u and $\tilde{u}$, $sim(\tilde{u}, u)$, is essentially an inverse distance measure and is used as a weight, i.e., the more similar users c and $\tilde{u}$ are, the more weight rating $r_{\tilde{u},m}$ will carry in the prediction of $r_{\tilde{u},m}$.

The similarity between two users is based on their ratings of items that both users have rated. There are several popular approaches: Pearson correlation, cosine-based, and Hamming-based similarities.

We further compare the cosine-based and normalised Hamming-based similarities:

$$sim_{cos}(u, v) = \sum_{m \in \tilde{M}} r_{um} \cdot r_{vm} / \left(\sum_{m \in \tilde{M}} r_{um}^2 \sum_{m \in \tilde{M}} r_{vm}^2 \right)^{1/2} \tag{4}$$

$$sim_{Ham}(u, v) = 1 - \sum_{m \in \tilde{M}} |r_{um} - r_{vm}| / |\tilde{M}|, \tag{5}$$

where $\tilde{M}$ is either the set of co-rated items (movies) for users u and v or the whole set of items.

To apply this approach in case of FCA-based BMF recommender algorithm we simply consider the user-factor matrices obtained after factorisation of the initial data as an input.

For the input matrix in Table 1 the corresponding decomposition is below:

$$\begin{bmatrix} 1&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&1&1&0&0 \\ 1&0&0&0&0&1&0&1&1 \\ 0&1&0&0&0&1&1&0&0 \\ 0&0&1&0&1&0&0&0&0 \\ 1&0&0&1&0&0&0&1&1 \\ 1&0&1&0&1&0&0&0&0 \\ 1&0&0&0&1&0&0&1&0 \\ 0&0&0&0&1&0&1&0&0 \\ 1&0&0&0&0&0&0&0&0 \end{bmatrix} \circ \begin{bmatrix} 1&0&0&1&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&1&0 \\ 0&0&0&0&1&1&1&0&0&0&1 \\ 1&0&1&1&0&0&0&1&1&0&0 \\ 0&0&0&0&1&1&0&0&0&0&0 \\ 0&0&1&0&0&1&0&0&0&1&0 \\ 0&1&1&0&0&1&0&0&0&0&0 \\ 1&0&1&1&0&0&0&0&0&0&0 \\ 1&0&1&1&0&0&0&1&0&0&0 \end{bmatrix}$$

3 Proposed Approach

In contrast to [5], for the recommender setting we mostly interested whether the concepts of more balanced extent and intent size may give us an advantage and use the following criterion to this end:

$$W(A,B) = (2|A||B|)/(|A|^2 + |B|^2) \in [0;1], \tag{6}$$

where (A, B) is a formal concept.

In subsection 2.2 we recalled that finding Boolean factors is reduced to the task of finding of covering formal concepts for the same input matrix.

To this end we modified *Close-by-One* ([13]). This algorithm traverses the tree of corresponding concept lattice in depth-first manner and returns the set of all formal concepts, which is redundant for the Boolean decomposition task. The deeper the algorithm is in the tree, the larger the intents are, and the smaller the extents of formal concepts. Thus, for every branch of the tree the proposed measure in eq. (6) is growing until some depth and then (in case the traverse continues) goes down.

The proposed modifications are: 1) the traverse of a certain branch is carried out until W is growing with the covered square (size of extent $\times$ size of intent); 2) at each iteration we do not accept concepts with intents that contained in the union of intents of previously generated concepts.

In case the intent of a certain concept is covered by its children (fulfilling condition 1), then this concept is not included into $\mathcal{F}$.

For *Close-by-One* there is a linear order $\succ$ on G. Assume $C \subset G$ is generated from $A \subset G$ by addition $g \in G$ ($C = A \cup \{g\}$) such that $g \succ \max(A)$, then the set C'' is called *canonically generated* if $min(C'' \setminus C) \succ g$.

Algorithm 1: Generation of balanced formal concepts

Data: Formal context (U, M, I)
Result: The set of balanced formal concepts $\mathcal{F}$
foreach $u \in U$ **do**
  $A \leftarrow \{u\}$;
  $stack.push(A')$;
  $g \leftarrow u$; $g++$;
  repeat
    if $g \notin U$ **then**
      if $stack.Top \neq \emptyset$ **then**
        add (A'', A') to $\mathcal{F}$;
        $stack.Top \leftarrow \emptyset$;
      while $stack.Top = \emptyset$ **do**
        $g \leftarrow \max(A)$;
        $A \leftarrow A \setminus \{g\}$;
        $stack.pop$;
        $g++$;
    else
      $B \leftarrow A \cup \{g\}$;
      if *(B'' is a canonical generation) and* $W(C'', C') \geq W(A'', A')$ *and* $|C'' \times C'| \geq |A'' \times A'|$ **then**
        $stack.Top \leftarrow (stack.Top \setminus C')$;
        $A \leftarrow C$;
      $g++$;
  until $A \neq \emptyset$;
return $\mathcal{F}$;

The obtained set $\mathcal{F}$ is still be redundant, that is why we further select factors with maximal coverage until we have covered the whole matrix or required percentage.

The main aim of factorisation is the reduction of computation steps and revealing latent similarity since users' similarities are computed in a factor space. As a projection matrix of user profiles to a factor space one may use *"user-factor"* from Boolean factorisation of utility matrix (P in (2)). However, in this case in the obtained user profiles most of the vector components are getting zeros, and thus we lose similarity information.

To smooth the loss effects we proposed the following weighted projection:

$$\tilde{P}_{uf} = \frac{I_{u\cdot} \cdot Q_{f\cdot}}{||Q_{f\cdot}||_1} = \frac{\sum\limits_{v \in V} I_{uv} \cdot Q_{fv}}{\sum\limits_{v \in V} Q_{fv}},$$

where $\tilde{P_{uf}}$ indicates whether factor f covers user u, $I_{u\cdot}$ is a binary vector describing profile of user u, $Q_{f\cdot}$ is a binary vector of items belonging to factor f

(the corresponding row of Q in decomposition eq. (2)). The coordinates of the obtained projection vector lie within $[0; 1]$.

For Table 1 the weighted projection is as follows:

$$\tilde{P} = \begin{bmatrix} 1 & \frac{1}{5} & 0 & 1 & 0 & \frac{1}{3} & \frac{1}{3} & 1 & 1 \\ 0 & 1 & \frac{1}{2} & \frac{1}{5} & \frac{1}{2} & 1 & 1 & \frac{1}{3} & \frac{1}{4} \\ 1 & \frac{3}{5} & \frac{1}{4} & \frac{4}{5} & \frac{1}{2} & 1 & \frac{2}{3} & 1 & 1 \\ 0 & 1 & \frac{1}{2} & \frac{1}{5} & \frac{1}{2} & 1 & 1 & \frac{1}{3} & \frac{1}{4} \\ 0 & \frac{2}{5} & 1 & 0 & 1 & \frac{2}{3} & \frac{1}{3} & 0 & 0 \\ 1 & \frac{1}{5} & 0 & 1 & 0 & \frac{1}{3} & \frac{1}{3} & 1 & 1 \\ 1 & \frac{2}{5} & 1 & \frac{1}{5} & 1 & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{2} \\ 1 & \frac{2}{5} & \frac{1}{2} & \frac{2}{5} & 1 & \frac{2}{3} & \frac{2}{3} & 1 & \frac{3}{4} \\ 0 & \frac{2}{5} & \frac{1}{2} & \frac{1}{5} & 1 & \frac{2}{3} & 1 & \frac{1}{3} & \frac{1}{4} \\ 1 & 0 & 0 & \frac{1}{5} & 0 & 0 & 0 & \frac{2}{3} & \frac{1}{2} \end{bmatrix}.$$

4 Experiments

The proposed approach and compared ones have been implemented in C++ and evaluated on the MovieLens-100k data set. This data set features 100000 ratings in five-star scale, 1682 Movies, Contextual information about movies (19 genres), 943 users (each user has rated at least 20 movies), and demographic info for the users (gender, age, occupation, zip (ignored)). The users have been divided into seven age groups: under 18, 18-25, 26-35, 36-45, 45-49, 50-55, 56+.

Five star ratings are converted to binary scale by the following rule:

$$I_{ij} = \begin{cases} 1, & R_{ij} > 3, \\ 0, & \text{else} \end{cases}$$

The scaled dataset is split into two sets according to bimodal cross-validation scheme [16]: training set and test set with a ratio 80:20, and 20% of ratings in the test set are hidden[3].

Measure of users similarity First of all, the influence of similarity has been compared. As we can see in the Fig. 4, Hamming distance based similarity is significantly better in terms of MAE and Precision. However it is worse in Recall and F-measure. Even though, given the superiority in terms of MAE (widely adopted in the RS community measure), we decided to use Hamming distance based similarity.

Projection into factor space In the series of tests the influence of projection method has been studied. The weighted projection keeps more information and as a result helps us to find similar user of higher accuracy. That is why this method has significant primacy in terms of all investigated measures of quality.

[3] This partition into test and training set is done 5 times resulting in 25 hidden submatrices and differs from the one provided by MovieLens group; hence the results might be different.

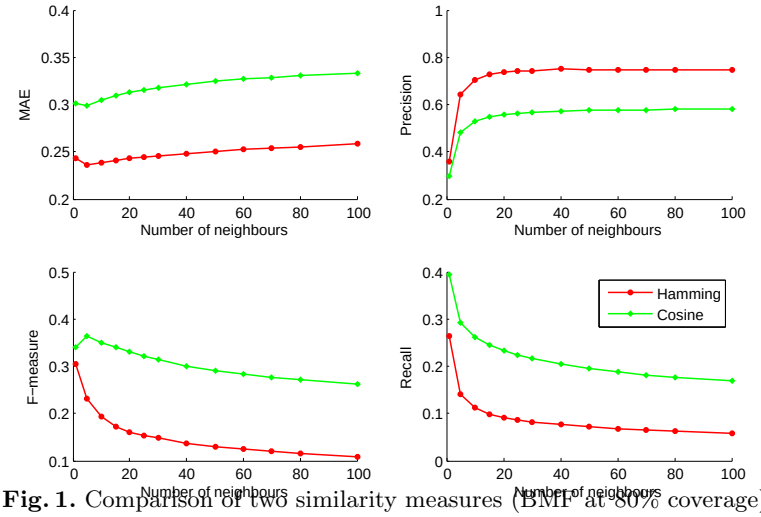


Fig. 1. Comparison of two similarity measures (BMF at 80% coverage)

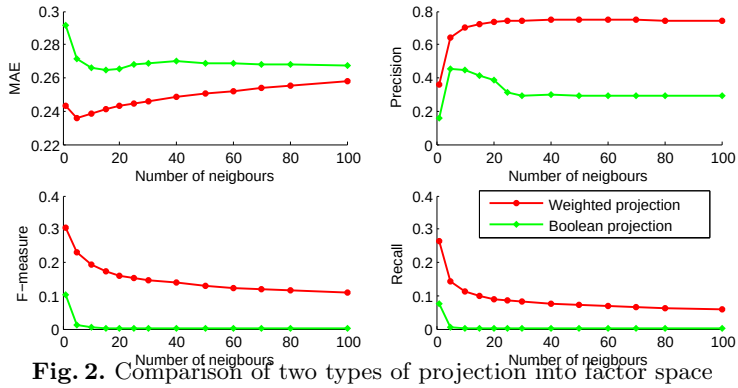


Fig. 2. Comparison of two types of projection into factor space

FCA-based algorithm and factors number The main studied algorithm to find Boolean factors as formal concepts is a modified algorithm *Close by One*. It was compared with greedy algorithm from [5] in terms of factors number and final RS quality measures.

Coverage	50%	60%	70%	80%	90%
Modified Close by One	168	228	305	421	622
Greedy algorithm	222	297	397	533	737

CbO covers the input matrix with a smaller count of factors, but it requires more time (in our experiments, 180 times more on average with one thread calculations). At the same time we have to admit that there is no influence to RS quality: thus Recall, Precision and MAE mainly differ only in the third digit.

Incorporation of context information and comparison with SVD For the SVD-based approach additional (context) information has been attached in a similar

way, but there we use maximal rating (5 stars) in the attached columns and rows.

Coverage	50%	60%	70%	80%	85%	90%
BMF	168	228	305	421	508	622
BMF (No context information)	163	220	294	401	479	596
SVD	162	218	287	373	430	496
SVD (No context information)	157	211	277	361	416	480

BMF and SVD give similar number of factors, especially for small coverage; context information does not significantly change their number, but it gives an increase of precision (1-2% more accurate predictions in Table 4).

Table 2. Influence of contextual information (80% coverage)

Number of neighbours	Precision		Recall		F-measure		MAE	
	clean	cntxt	clean	cntxt	clean	cntxt	clean	cntxt
1	0.3589	0.3609	0.2668	0.2647	0.3061	0.3054	0.2446	0.2434
5	0.6353	0.6442	0.1420	0.1412	0.2321	0.2317	0.2371	0.2359
10	0.6975	0.7045	0.1126	0.1114	0.1938	0.1924	0.2399	0.2388
15	0.7168	0.7258	0.0994	0.0979	0.1746	0.1726	0.2422	0.2411
20	0.7282	0.7373	0.0911	0.0903	0.1619	0.1610	0.2442	0.2429
25	0.7291	0.7427	0.0861	0.0853	0.1540	0.1531	0.2457	0.2445
30	0.7318	0.7426	0.0823	0.0818	0.1480	0.1474	0.2472	0.2459
40	0.7342	0.7508	0.0767	0.0759	0.1389	0.1379	0.2497	0.2484
50	0.7332	0.7487	0.0716	0.0712	0.1304	0.1301	0.2518	0.2504
60	0.7314	0.7478	0.0682	0.0678	0.1247	0.1243	0.2536	0.2522
70	0.7333	0.7477	0.0658	0.0654	0.1208	0.1202	0.2552	0.2538
80	0.7342	0.7449	0.0632	0.0624	0.1164	0.1151	0.2567	0.2553
100	0.7299	0.7461	0.0590	0.0583	0.1092	0.1081	0.2594	0.2580

With a similar number or factors (SVD at 85% coverage and BMF at 80%) Boolean Factorisation results in smaller MAE and higher Precision where number of neighbours is not high. It can be explained by different nature of factors in these factorisation models.

5 Conclusion

In the paper we considered two modifications of Boolean matrix factorisation, which are suitable for Recommender Systems. They were compared on real datasets with the presence of auxiliary (context) information. We found out that MAE of our BMF-based approach is sufficiently lower than MAE of SVD-based approach for almost the same number of factor at fixed coverage level of BMF and SVD. The Precision of BMF-based approach is slightly lower when the number of neighbours is about a couple of dozens and comparable for the

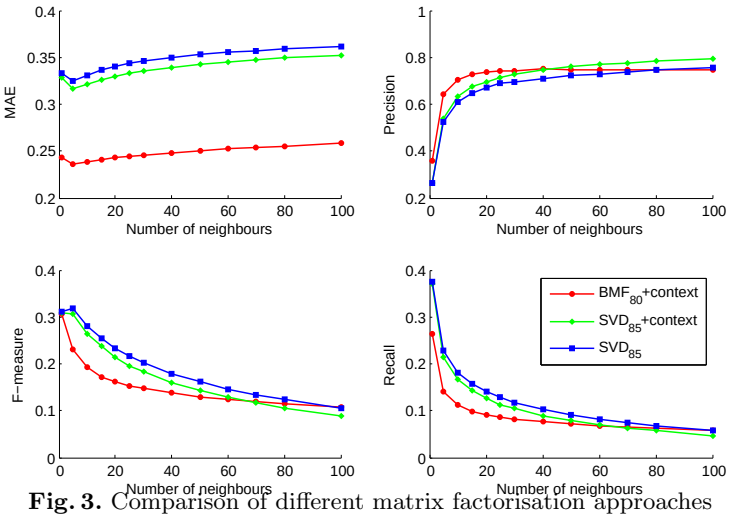


Fig. 3. Comparison of different matrix factorisation approaches

remaining part of the observed range. The Recall is lower than results in lower F-measure. The proposed weighted projection alleviates the information loss of original Boolean projection resulting in a substantial quality gain.

We also revealed that the presence of contextual information results in a small quality increase (about 1-2%) in terms of MAE, Recall and Precision.

We studied the influence of more balanced factors in terms of ratio of number of users and items. Finally, we should report that greedy approximate algorithm [5], even though that it results in more factors with larger user's component, is faster and demonstrates almost the same quality. So, its use is beneficial for recommender systems due to polynomial time computational complexity.

As a future research direction we would like to investigate the proposed approach with the previously ([9,6,10,7]) and recently introduced FCA-based ones ([11,12,17]). As for Boolean matrix factorisation in case of context-aware information, since the data can be naturally represented as multi-relational, we would like to continue our collaboration with the authors of the paper [18]. We definitely need to use user- and item-based independent information like time and location, which can be considered as pure contextual in nature and treated by n-ary methods [19].

Acknowledgments. We would like to thank Alexander Tuzhilin, Elena Nenova, Radim Belohlavek, Vilem Vychodil, Sergei Kuznetsov, Sergei Obiedkov, Vladimir Bobrikov, Mikhail Roizner, and anonymous reviewers for their comments, remarks and explicit and implicit help during the paper preparations. This work was supported by the Basic Research Program at the National Research University Higher School of Economics in 2014-2015 and performed in the Laboratory of Intelligent Systems and Structural Analysis. First author was also supported by Russian Foundation for Basic Research (grant #13-07-00504).

References

1. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8) (2009) 30–37
2. Trefethen, L.N., Bau, D.: Numerical Linear Algebra. 3rd edition edn. SIAM (1997)
3. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine Learning **42**(1-2) (2001) 177–196
4. Lin, C.J.: Projected gradient methods for nonnegative matrix factorization. Neural Comput. **19**(10) (October 2007) 2756–2779
5. Belohlavek, R., Vychodil, V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. Journal of Computer and System Sciences **76**(1) (2010) 3 – 20 Special Issue on Intelligent Data Analysis.
6. Symeonidis, P., Nanopoulos, A., Papadopoulos, A., Manolopoulos, Y.: Nearest-biclusters collaborative filtering based on constant and coherent values. Information Retrieval **11**(1) (2008) 51–75
7. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J.: Concept-Based Biclustering for Internet Advertisement. In: Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on. (Dec 2012) 123–130
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin/Heidelberg (1999)
9. du Boucher-Ryan, P., Bridge, D.: Collaborative recommending using formal concept analysis. Knowledge-Based Systems **19**(5) (2006) 309 – 315 {AI} 2005 {SI}.
10. Ignatov, D.I., Kuznetsov, S.O.: Concept-based recommendations for internet advertisement. In Belohlavek, R., Kuznetsov, S.O., eds.: Proc. of The Sixth International Conference Concept Lattices and Their Applications (CLA'08), Palacky University, Olomouc (2008) 157–166
11. Alqadah, F., Reddy, C., Hu, J., Alqadah, H.: Biclustering neighborhood-based collaborative filtering method for top-n recommender systems. Knowledge and Information Systems (2014) 1–17
12. Ignatov, D.I., Nenova, E., Konstantinova, N., Konstantinov, A.V.: Boolean Matrix Factorisation for Collaborative Filtering: An FCA-Based Approach. In: Artificial Intelligence: Methodology, Systems, and Applications - 16th Int. Conf., AIMSA 2014, Varna, Bulgaria, September 11-13, 2014. Proceedings. (2014) 47–58
13. Kuznetsov, S.O.: A fast algorithm for computing all intersections of objects in a finite semilattice. Automatic Documentation and Math. Ling. **27**(5) (1993) 11–21
14. Birkhoff, G.: Lattice Theory. 11th edn. Harvard University, Cambridge, MA (2011)
15. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. on Knowl. and Data Eng. **17**(6) (June 2005) 734–749
16. Ignatov, D.I., Poelmans, J., Dedene, G., Viaene, S.: A New Cross-Validation Technique to Evaluate Quality of Recommender Systems. In Kundu, M., Mitra, S., Mazumdar, D., Pal, S., eds.: Perception and Machine Intelligence. Volume 7143 of LNCS. Springer (2012) 195–202
17. Ignatov, D.I., Kornilov, D.: Raps: A recommender algorithm based on pattern structures. In: Proceeding of FCA4AI 2015 workshop at IJCAI 2015. (2015)
18. Trnecka, M., Trneckova, M.: An algorithm for the multi-relational boolean factor analysis based on essential elements. In: Proceedings of 11th International Conference on Concept Lattices and their Applications. (2014)
19. Ignatov, D., Gnatyshak, D., Kuznetsov, S., Mirkin, B.: Triadic formal concept analysis and triclustering: searching for optimal patterns. Machine Learning (2015) 1–32

Class Model Normalization

Outperforming Formal Concept Analysis approaches with AOC-posets

A. Miralles[1,2], G. Molla[1], M. Huchard[2], C. Nebut[2],
L. Deruelle[3], and M. Derras[3]

(1) Tetis/IRSTEA, France
andre.miralles@teledetection.fr, guilhem.molla@irstea.fr
(2) LIRMM, CNRS & Université de Montpellier, France
huchard,nebut@lirmm.fr
(3) Berger Levrault, France
laurent.deruelle@berger-levrault.com,mustapha.derras@berger-levrault.com

Abstract. Designing or reengineering class models in the domain of programming or modeling involves capturing technical and domain concepts, finding the right abstractions and avoiding duplications. Making this last task in a systematic way corresponds to a kind of model normalization. Several approaches have been proposed, that all converge towards the use of Formal Concept Analysis (FCA). An extension of FCA to linked data, Relational Concept Analysis (RCA) helped to mine better reusable abstractions. But RCA relies on iteratively building concept lattices, which may cause a combinatorial explosion in the number of the built artifacts. In this paper, we investigate the use of an alternative RCA process, relying on a specific sub-order of the concept lattice (AOC-poset) which preserves the most relevant part of the normal form. We measure, on case studies from Java models extracted from Java code and from UML models, the practical reduction that AOC-posets bring to the normal form of the class model.

Keywords: Inheritance hierarchy, class model normalization, class model reengineering, Formal Concept Analysis, Relational Concept Analysis

1 Introduction

In object-oriented software or information systems, the specialization-generalization hierarchy is a main dimension in class model organization, as the *is-a* relation in the design of domain ontologies. Indeed, it captures a classification of the domain objects which is structuring for human comprehension and which makes the representation efficient. Designing or reengineering class models in the domain of programming or in the domain of modeling still remains a tricky task. It includes the integration of technical and domain concepts sometimes with no clear semantics, and the definition of the adequate abstractions while avoiding the duplication of information. In many aspects, this task corresponds to a

kind of class model normalization, focusing on specialization and redundancy, by analogy to the database schema normalization. Normalization is important to assist forward engineering of a reliable and maintainable class model. It is also useful to address the erosion of the specialization-generalization hierarchy during software evolution.

After the seminal paper of R. Godin and H. Mili at OOPSLA'93 [1], several approaches have been proposed to address this normalization, that all converged to the implicit or explicit use of Formal Concept Analysis (FCA [2]) techniques. In this context, FCA was used to mine descriptions that are common to groups of classes and to suggest re-factorizing and creating more reusable super-classes. Several approaches more specifically used a specific sub-order of the concept lattice which captures the most relevant new super-classes (the AOC-poset, for Attribute-Object Concept poset [3]). Then, Relational Concept Analysis (RCA [4]), an extension of FCA to linked data, was proposed to find deeper re-factorizations. However, RCA iterates on building concept lattices, that leads to a combinatorial explosion of the number of the built artifacts (including classes).

In this paper, we investigate the use of an alternative version of RCA, relying on AOC-posets. With AOC-posets, RCA might not converge, thus we have to carefully handle the iteration mechanism, but when it converges, we expect more efficiency. We measure, on case studies from UML models and from Java models rebuilt from industrial Java code, the reduction brought in practice by this approach to the normal form of the class model. We also show that, on realistic tuning, the reasonable number of the new artifacts allows the designer to analyze them and decide which of them should be kept in the model.

In Section 2, the bases for FCA and RCA in the context of class models are outlined. Then we present current work in this domain, and the motivation for this study (Section 3). In Section 4, we give the setup of the experiment, as well as the results. We conclude in Section 5 with a few perspectives of this work.

2 Class Model Normalization

FCA: A classical approach for applying FCA to class model normalization involves building a formal context **K-class**$=(G, M, I)$, where the classes (in G) are associated with their characteristics (attributes, roles, operations, in M) through $I \subseteq G \times M$. There are many variations in this description of classes. For example, Fig. 1 (right-hand side) shows such a formal context for the class model presented in Fig. 2(a). A *concept* is a pair $(Extent, Intent)$ where $Extent = \{g \in G | \forall m \in Intent, (g, m) \in I\}$ and $Intent = \{m \in M | \forall g \in Extent, (g, m) \in I\}$. The concept extent represents the objects that own all the characteristics of the intent; the concept intent represents the characteristics shared by all objects of the extent. The specialization order between two formal concepts is given by: $(Extent\_1, Intent\_1) < (Extent\_2, Intent\_2) \Leftrightarrow Extent\_1 \subset Extent\_2$. It provides the concepts with a lattice structure.

In the concept lattice, there is an ascending inheritance of objects and a descending inheritance of characteristics. The simplified intent of a formal concept

Device	recommendedHeight	windVaneDimension	cupNumber	vaneType	plateDimension	tubeLength
CupAnemometer	×		×			
VaneAnemometer	×	×		×		
PlateAnemometer	×	×			×	
PitotAnemometer		×				×

K-class	tubeHeight	measureInterval	accuracy	measuringDate	codeQuality	waterAmount	windStrength	windDirection	rainfall	wind
RainGauge	×								×	
Anemometer		×	×							×
Rainfall				×	×	×				
Wind				×	×		×	×		

Fig. 1. Anemometer Formal Context (left-hand side), Formal context **K-class**=(G, M, I) with G is the set of classes of Fig. 2(a) associated by I with the set M of their attribute and role names (right-hand side)

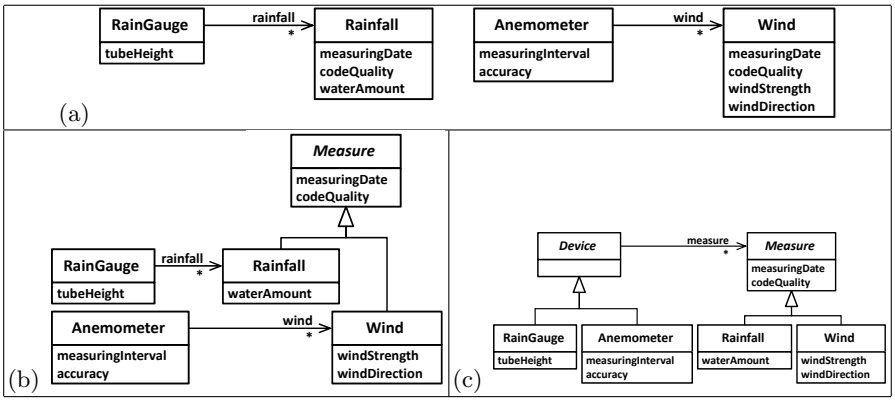


Fig. 2. Example of class model normalization with FCA and RCA [5]: (a) initial class model ; (b) (resp. (c)) class model refactored with FCA (resp. RCA).

is its intent without the characteristics inherited from its super-concept intents. The simplified extent is defined in a similar way. In our example, among the formal concepts that can be extracted from **K-class**, Concept C = (`{Rainfall, Wind}`, `{measuringDate, codeQuality}`) highlights two classes that share the two attributes `measuringDate` and `codeQuality`. This concept C is interpreted as a new super-class of the classes of its extent, namely `Rainfall` and `Wind`. The new super-class, called here `Measure`, appears in Fig. 2(b). New super-classes are named by the class model designer.

AOC-posets: In the framework of class model analysis with FCA, often AOC-posets, rather than concept lattices, are used. Formal context of Fig. 1 (left-hand side) is used to illustrate the difference between the concept lattice and the AOC-poset. The concept lattice like in Fig. 3(a) contains all the concepts from the formal context. Some concepts, like `Concept_Device_2`, inherit all their characteristics from their super-concepts and their objects from their sub-concepts. In the particular case of object-oriented modeling, they would correspond to empty

description, with no correspondence with an initial class description and be rarely considered. In the AOC-poset like in Fig. 3(b), only concepts that introduce one characteristic or one object are kept, simplifying drastically the structure in case of large datasets. The number of concepts in the concept lattice can increase up to $2^{min(|G|,|M|)}$, while it is bounded by $|G|+|M|$ in the AOC-poset. The Iceberg lattice, such as introduced in [6], is another well known sub-set of the concept lattice which is used in many applications. The iceberg lattice is induced by the sub-set of concepts which have an extent support greater than a given threshold. In our case, this would mean only keeping new super-classes that have a minimum number of sub-classes, which is not relevant in modeling and programming: a super-class may only have one sub-class.

RCA: RCA helps to go further and get the class model of Fig. 2(c). In this additional normalization step, `RainGauge` and `Anemometer` have a new super-class which has been discovered because both have a role towards a sort of `Measure` (resp. `Rainfall` and `Wind`). To this end, the class model is encoded in a Relational Context Family (RCF) as the one in Table 1, composed of several formal contexts that separately describe classes (`K-class`), attributes (`K-attribute`), and roles (`K-role`) and of several relations including relation between classes and attributes (`r-hasAttribute`), relation between classes and roles (`r-hasRole`), or relation between roles and their type `r-hasTypeEnd`. Here again, this encoding can vary and integrate other modeling artifacts, like operations or associations.

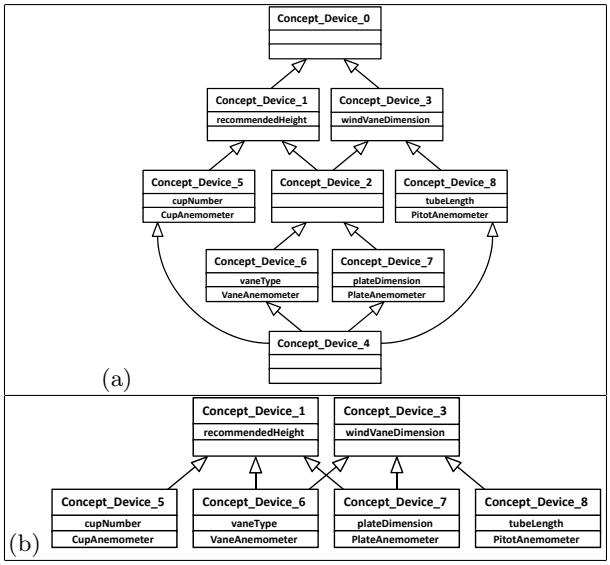


Fig. 3. Concept lattice (a) and AOC-poset (b) for anemometers (Fig. 1, left) [5]

RCA is an iterative process where concepts emerge at each step. Relations and concepts discovered at one step are integrated into contexts through relational attributes for computing concept lattices at the next step. At step 0, attributes with the same name (resp. the two attributes `measuringDate` or the two attributes `codeQuality`) are grouped. At step 1, classes that share attributes from an attribute group are grouped into a concept that produces a new super-class (e.g. `Wind` and `Rainfall` are grouped to produce the super-class `Measure`). At step 2, roles `rainfall` and `wind` share the fact that they end at a sub-class of `Measure`, thus they are grouped into new role shown under the name `measure` in Fig. 2(c). At step 3, the current context of classes (extended with relational attributes) shows that both classes `RainGauge` and `Anemometer` have a role ending to `Measure`. Then a new super-class, called `Device` by the designer, is extracted.

Table 1. Context family for the set of classes of Fig. 2(a)

Krole	rainfall	wind
rainfall	×	
wind		×

Kattribute	tubeHeight	measureInterval	accuracy	measuringDate	codeQuality	waterAmount	windStrength	windDirection
RG::tubeHeight	×							
A::measureInterval		×						
A::accuracy			×					
R::measuringDate				×				
W::measuringDate				×				
R::codeQuality					×			
W::codeQuality					×			
R::waterAmount						×		
W::windStrength							×	
W::windDirection								×

Kclass
RainGauge
Anemometer
Rainfall
Wind

$r_{hasAttribute}$	RG::tubeHeight	A::measureInterval	A::accuracy	R::measuringDate	W::measuringDate	R::codeQuality	W::codeQuality	R::waterAmount	W::windStrength	W::windDirection
RainGauge	x									
Anemometer		x	x							
Rainfall				x		x		x		
Wind					x		x		x	x

$r_{hasRole}$	rainfall	wind
RainGauge	x	
Anemometer		x
Rainfall		
Wind		

$r_{hasTypeEnd}$	RainGauge	Anemometer	Rainfall	Wind
rainfall			x	
wind				x

3 Previous work on RCA and Class Model Normalization

RCA has been first assessed on small [7] or medium [8] class models, encoding technical information (multiplicity, visibility, being abstract, initial value) in the RCF, which was the source of many non-relevant concepts.

In [9], the authors assessed RCA on Ecore models, Java programs and UML models. The encoding was similar to the one presented in Section 2 to illustrate RCA (classes, attributes, roles, described by their names and their relationships).

While for Java models, the number of discovered class concepts (about 13%) was very reasonable, for UML class models, the increase (about 600%) made the post-analysis impossible to achieve.

Recently, we systematically studied various encodings of the design class model of an information system about Pesticides [10, 11]. We noticed a strong impact of association encoding, and that encoding only named and navigable ends of associations was feasible, while encoding all ends (including those without a name and those that are not navigable) led to an explosion of the number of concepts. Restricting to named ends and to navigable ends means that we give strong importance to the semantics used by designer, thus the lost concepts have a greater chance to be uninteresting.

Guided by the intuition of the model designer, we recently proposed a controlled approach with progressive concept extraction [5]. In this approach, the designer chooses at each step of the RCA process the formal contexts and the relations he wants to explore. For example, he may begin with classes and attributes, then add roles, then associations, then remove all information about classes and consider only classes and operations, etc. Such a choice is memorized in a factorization path. In [5], we used AOC-posets, but we did not evaluate the difference between AOC-posets and concept lattices in the controlled process. The objective was to evaluate the number of discovered concepts at each step and to observe trends in their progress. It was worth noting that the curves of the same factorization path applied to different models had the same shape.

In this paper, we will use the 15 versions of the analysis class model of the same information system on Pesticides, as well as a dataset coming from industrial Java models. Contrarily to the experiments made by authors in [9–11], our objective is to evaluate the benefits of using the variant of RCA, which builds AOC-posets (rather than concept lattices) during the construction process. Studying the concepts discovered at each step, we can also evaluate what was called the *automatic factorization path* in the controlled approach of [5]. It is clear that AOC-posets are smaller than concept lattices, thus the results we expect focus on assessing the amount of the reduction in the number of discovered concepts that will be brought to the designer for analysis.

4 Case study

Experimental setup: Figure 4 presents the metamodel used in practice to define the RCF for our experiments. Object-Attribute contexts are associated to classes, attributes, operations, roles and associations. Attributes, operations, roles and associations are described by their names in UML and Java models. Classes are described by their names in UML models.

Object-object contexts correspond to the meta-relations `hasAttribute`, `hasOperation`, `hasRole` (between `Class` and `Role` and between `Association` and `Role`), `hasTypeEnd`, and `isRoleOf`. This last meta-relation is not used in the Java model RCF, because from Java code we can only extract unidirectional associations (corresponding to Java attributes whose type is a class).

All the roles extracted from Java code have a name. In UML models, we only consider the named roles, and the navigable ends of associations to focus on the most meaningful elements. We do not consider multiplicities in role description, nor the initializer methods (constructors).

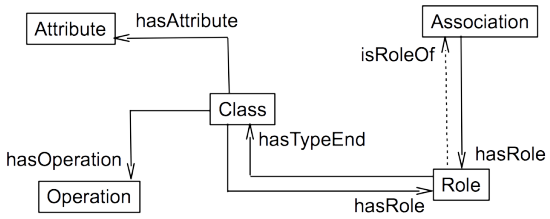


Fig. 4. Metamodel used to define the RCF.

Each of the 15 UML models corresponds to a version of an information system on Pesticides. These models, described in [10], were collected during the Pesticides information system development and then the evolution of the project throughout 6 years. The RCF for UML models contains all the UML model elements. We also used 15 Java models coming from Java programs developed by the company Berger-Levrault in the domain of human resource management. These 15 Java models come from 3 Java programs: Agents, Chapter and Appraisal Campaign. For each program, we determined a central meaningful class and navigated from this central class through association roles at several distances: 1, 2, 4, 8 and 16. For the 5 Java models, we could not get results due to the size of the models. The Java programs are the Java counterpart of the database accesses, thus we focused on Java attributes, that are encoded as attributes when their type is primitive (integer, real, boolean, etc), and as roles of a unidirectional associations when their type is a class. The operations were access and update operations associated to these attributes, thus they do not bring any meaningful information. For the sake of space, we only present results on 4 representative Java models, from the `Chapter` program (distances 1, 2, 4, 8). Depending on the version, 254 to 552 model elements are involved in the Pesticides models and 34 to 171 of these model elements are classes. The Chapter models are composed of 204 to 3979 model elements of which 37 to 282 are classes.

The experiments have been made with the help of UML profiles of Objecteering[1] to extract the RCF from the UML models and Java modules to extract the RCF from the Java models. RCAExplore[2] is used to compute the lattices and

[1] www.objecteering.com/
[2] dolques.free.fr/rcaexplore/

the AOC-posets, and Talend[3] to extract information from RCAExplore outputs, to integrate data, to build the curves and analyze the results.

Results: We computed various metrics on the built lattices and AOC-posets, such as the number of several categories of concepts: merged concepts (simplified extents have a cardinal strictly greater than 1), perennial concepts (simplified extents have a cardinal equals to 1) and new concepts (simplified extents are empty). A merged concept corresponds to a set of model elements that have the same description, for example a merged attribute concept groups attributes with a same name. A perennial concept corresponds to a model element which has at least one characteristic that makes it distinct from the others. A new concept corresponds to a group of a model elements that share part of their description, and no model element has exactly this description (it always has some additional information). We focus in this paper on the classes and on the number of new class concepts, because they are the predominating elements that reveal potential new abstractions and a possible lack of factorization in the current model. To highlight the observed increase brought by the conceptual structures (lattice and AOC-poset), we present the ratio of new class concepts on the number of initial classes in the model (Fig. 5, 6, 7, 8). To compare lattices and AOC-posets, we compute the ratio: $\frac{\#New\ class\ concepts\ in\ lattice}{\#New\ class\ concepts\ in\ AOC-poset}$ (Fig. 9, 10). For Chapter models, lattices are computed for the steps 0 to 6 and, for all other cases, lattices or AOC-posets are determined up to step 24.

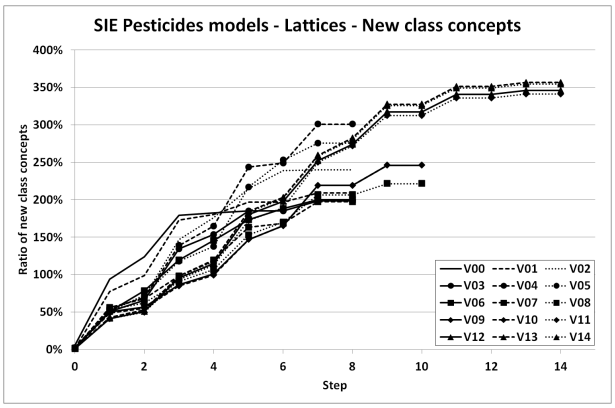


Fig. 5. New class concept number in lattices for Pesticides models

In lattices for the Pesticides models (Fig. 5), the process always stops between steps 6 and 14 depending on the models. At step 6, the ratio of new class concepts on the number of classes varies between 165% (V10) and 253% (V05). Results

[3] www.talend.com/

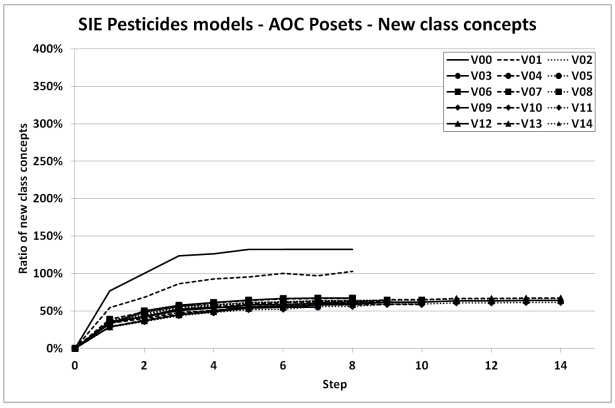


Fig. 6. New class concept number in AOC-posets for Pesticides models

are hard to analyze for a human designer without any further filtering. In lattices for the Chapter models (Fig. 7), we stopped the process at step 6, because we observed a high complexity: for example, for distance 8, at step 6, we get 43656 new class concepts. It is humanly impossible to analyze the obtained new class concepts. At distance 16, Chapter model could not even be processed. For the model at distance 1 (resp. 2), at step 6, the ratio is 11 % (resp. 62%) remaining reasonable (resp. beginning to be hard to analyze). We also observe stepping curves, that are explained by the metamodel: at step 1, new class concepts are introduced due to attribute and role concept creation of step 0, then they remain stable until step 2, while role and association concepts increase, etc...

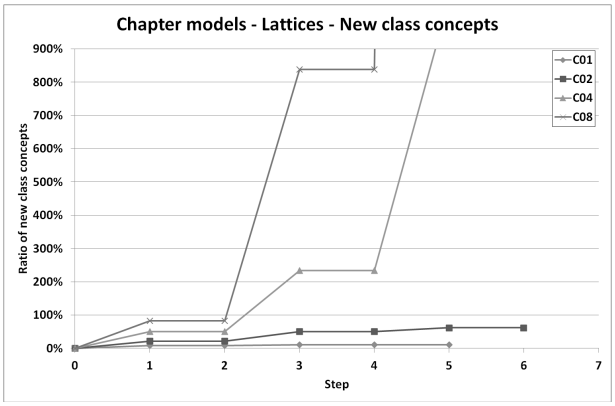


Fig. 7. New class concept number in lattices for Chapter models

Curves for AOC-posets for Pesticides models are shown in Fig. 6. The ordering on Pesticides AOC-posets roughly corresponds to the need of factorization: the highest curves correspond to V00 and V01 where few inheritance relations (there is none in V00) have been used. This shows the factorization work done by the designer during model versioning. The ratio at step 6 varies between 56% (V10) and 132% (V00). In Pesticides lattices, we observed many curve crossings, and curves have different shapes, while with Pesticide AOC-posets, the curves have a regular shape and globally they decrease.

For Chapter models (Fig. 8), convergence is obtained for all AOC-posets (distance 1 to 8) and the process stops between steps 5 and 23. The curves are also ordered and, as for lattices, the highest curve corresponds to the highest distance. The curves reveal many opportunities for factorization. The ratio at step 6 varies between 5% (distance 1) and 161% (distance 8).

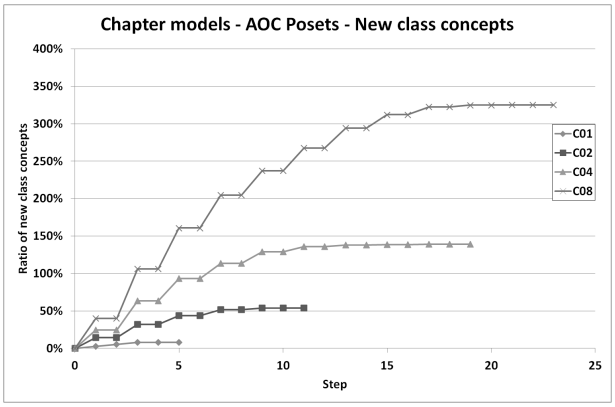


Fig. 8. New class concept number in AOC-posets for Chapter models

Fig. 9 and 10 allow lattices and AOC-posets to be compared. We always have more concepts in lattices than in AOC-posets, which was expected. In the AOC-poset of the Chapter model at distance 1, there is only one new class concept, while in the lattice they are three (including the top and bottom concepts), explaining the beginning of the curve (Fig. 10). For version V00, the behavior of the lattice-based process and the AOC-based process are similar. Except for version V04 (at version V05 a duplication of a part of the model has been made for working purpose), the highest curves correspond to the last models, which have been better factorized, and we here notice the highest difference between the two processes. We may hypothesize that lattices may contain many uninteresting factorizations compared to AOC-posets in these cases. This experiment shows that, in practice, the AOC-posets generally produce results that can be analyzed, while lattices are often difficult to compute in some cases and are often too huge to be used for our purpose.

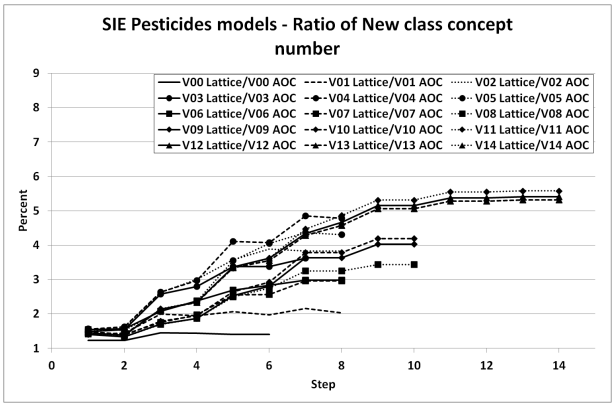


Fig. 9. Ratio $\frac{\#New\ class\ concepts\ in\ lattice}{\#New\ class\ concepts\ in\ AOC-poset}$ for Pesticides models

5 Conclusion

For class model normalization, concept lattices and AOC-posets are two structures giving two different normal forms. UML models that are rebuilt from these structures are interesting in both cases from a thematic point of view. Nevertheless, lattices are often too huge, and AOC-posets offer a good technique to reduce the complexity.

As future work, we plan to go more deeply into an exploratory approach, defining different factorization paths, with model rebuilding at each step with expert validation. This would allow the complexity to be controlled introducing less new concepts at each step. We also plan to use domain ontologies to guide acceptance of a new formal concept, because it corresponds to a thematic concept.

Acknowledgment

This work has been supported by Berger Levrault. The authors also warmly thank X. Dolques for the RCAExplore tool which has been used for experiments.

References

1. Godin, R., Mili, H.: Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices. In: Proceedings of the Eight Annual Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 93), ACM (1993) 394–410
2. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundation. Springer-Verlag Berlin (1999)

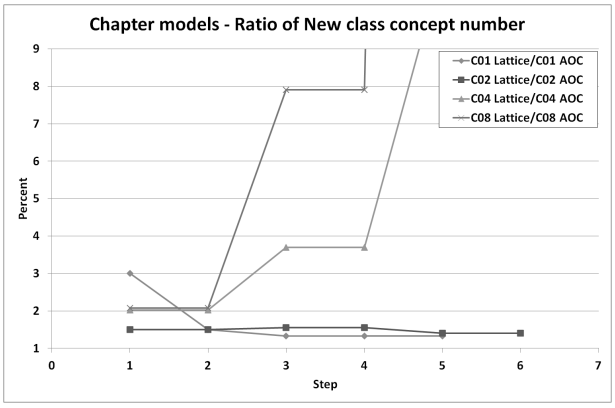


Fig. 10. Ratio $\frac{\#New\ class\ concepts\ in\ lattice}{\#New\ class\ concepts\ in\ AOC-poset}$ for Chapter models

3. Dolques, X., Le Ber, F., Huchard, M.: AOC-Posets: a Scalable Alternative to Concept Lattices for Relational Concept Analysis. In: Proceedings of the Tenth International Conference on Concept Lattices and Their Applications (CLA 2013). Volume 1062 of CEUR Workshop Proceedings., CEUR-WS.org (2013) 129–140
4. Rouane-Hacène, M., Huchard, M., Napoli, A., Valtchev, P.: Relational concept analysis: mining concept lattices from multi-relational data. Ann. Math. Artif. Intell. **67**(1) (2013) 81–108
5. Miralles, A., Huchard, M., Dolques, X., Le Ber, F., Libourel, T., Nebut, C., Guédi, A.O.: Méthode de factorisation progressive pour accroître l'abstraction d'un modèle de classes. Ingénierie des Systèmes d'Information **20**(2) (2015) 9–39
6. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing Iceberg Concept Lattices with TITANIC. Data Knowl. Eng. **42**(2) (2002) 189–222
7. Rouane-Hacène, M.: Relational concept analysis, application to software reengineering. Thèse de doctorat, Université du Québec à Montréal (2005)
8. Roume, C.: Analyse et restructuration de hiérarchies de classes. Thèse de doctorat, Université Montpellier 2 (2004)
9. Falleri, J.R., Huchard, M., Nebut, C.: A generic approach for class model normalization. In: Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008). (2008) 431–434
10. Osman Guédi, A., Miralles, A., Huchard, M., Nebut, C.: A practical application of relational concept analysis to class model factorization: Lessons learned from a thematic information system. In: Proceedings of the Tenth International Conference on Concept Lattices and Their Applications (CLA 2013). Volume 1062 of CEUR Workshop Proceedings., CEUR-WS.org (2013) 9–20
11. Osman Guédi, A., Huchard, M., Miralles, A., Nebut, C.: Sizing the underlying factorization structure of a class model. In: Proceedings of the 17th IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2013). (2013) 167–172

Partial enumeration of minimal transversals of a hypergraph

Lhouari Nourine, Alain Quilliot and Hélène Toussaint

Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, France
{nourine, quilliot, helene.toussaint}@isima.fr

Abstract. In this paper, we propose the first approach to deal with enumeration problems with huge number of solutions, when interestingness measures are not known. The idea developed in the following is to partially enumerate the solutions, i.e. to enumerate only a representative sample of the set of all solutions. Clearly many works are done in data sampling, where a data set is given and the objective is to compute a representative sample. But, to our knowledge, we are the first to deal with sampling when data is given implicitly, i.e. data is obtained using an algorithm. The experiments show that the proposed approach gives good results according to several criteria (size, frequency, lexicographical order).

1 Introduction

Most of problems in data mining ask for the enumeration of all solutions that satisfy some given property [1, 10]. This is a natural process in many applications, e.g. marked basket analysis [1] and biology [2] where experts have to choose between those solutions. An enumeration problem asks to design an output-polynomial algorithm for listing without duplications the set of all solutions. An output-polynomial algorithm is an algorithm whose running time is bounded by a polynomial depending on the sum of the sizes of the input and output.

There are several approachs to enumerate all solutions to a given enumeration problem. Johnson *et al.* [13] have given a polynomial-time algorithm to enumerate all maximal cliques or stables of a given graph. Fredman and Khachiyan [7] have proposed a quasi-polynomial-time algorithm to enumerate all minimal transversal of an hypergraph. For enumeration problems the size of the output may be exponential in the size of the input, which in general is different from optimization problems where the size of the output is polynomially related to the size of the input. The drawback of the enumeration algorithms is that the number of solutions may be exponential in the size of the input, which is infeasible in practice. In data mining, some interestingness measures or constraints are used to bound the size of the output, e.g. these measures can be explicitly specified by the user [8]. In operation research, we use quality criteria in order to consider appropriate decision [21].

In this paper, we deal with enumeration problems with huge number of solutions, *when interestingness measures are not known*. This case happens when

the expert has no idea about data and knowledges that are looking for. The objective is to enumerate only a *representative sample of the set of all solutions*. Clearly many works are done in *data sampling*, where are given a data set and the objective is to compute a representative sample. To our knowledges, this idea is new for sampling when data is given implicitly, i.e. data is obtained using an algorithm. One can use the naive approach which first enumerates all the solutions and then applies sampling methods, which is not possible for huge number of solutions.

To evaluate our approach, we consider a challenging enumeration problem, which is related to mining maximal frequent item sets [1, 10], dualization of monotone boolean functions [5] and other problems [10]. We applied our approach to several instances of transversal hypergraphs [17, 20], and obtain good results.

2 Related works

Golovach *et al.* [9] have proposed an algorithm to enumerate all minimal dominating sets of a graph. First they generate maximal independent sets and then apply a flipping operation to them to generate new minimal dominating sets, where the enumeration of maximal independent sets is polynomial. Clearly, a relaxation of the flipping operation leads to a partial enumeration since the number of minimal dominating sets can be exponential in the number of minimal independent sets, e.g. cobipartie graphs. Jelassi *et al.*[12] and Raynaud *et al.*[19] have considered some kind of redundancy in hypergraphs like twin elements to obtain a concise representation. Their ideas can avoid the enumeration of similar minimal transversals of an hypergraph.

3 Transversal hypergraph enumeration

A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ consists of a finite collection $\mathcal{E}$ of sets over a finite set V. The elements of $\mathcal{E}$ are called *hyperedges*, or simply *edges*. An hypergraph is said simple if for any $E, E' \in \mathcal{E}$ $E \not\subseteq E'$. A *transversal* (or *hitting set*) of $\mathcal{H}$ is a set $T \subseteq V$ that intersects every edge of $\mathcal{E}$. A vertex x in a transversal T is said to be redundant if $T \setminus \{x\}$ is still a transversal. A transversal is *minimal* if it does not contain any redundant vertex. The set $\mathcal{T}$ of all minimal transversal of $\mathcal{H} = (V, \mathcal{E})$ constitutes together with V also a hypergraph $Tr(\mathcal{H}) = (V, \mathcal{T})$, which is called the *transversal hypergraph* of $\mathcal{H}$. We denote by $k = \sum_{E \in \mathcal{E}} \mid E \mid$.

Example 1. Consider the hypergraph $\mathcal{H} = (V, \mathcal{E})$: $V = \{1, 2, 3, 4, 5\}$ and $\mathcal{E} = \{E_1, E_2, E_3\}$ with $E_1 = \{1, 3, 4\}$, $E_2 = \{1, 3, 5\}$ and $E_3 = \{1, 2\}$. The set of all minimal transversals is $\mathcal{T} = \{\{1\}, \{2, 3\}, \{2, 4, 5\}\}$ and $k = 3 + 3 + 2 = 8$

Given a simple hypergraph $\mathcal{H} = (V, \mathcal{E})$, the transversal hypergraph enumeration problem concerns the enumeration without repetitions of $Tr(\mathcal{H})$. This problem has been intensively studied due to its link with several problems isuch as

data mining and learning [3, 4, 11, 15, 18]. Recently, Kante *et al.*[14] have shown that the enumeration of all minimal transversals of an hypergraph is polynomially equivalent to the enumeration of all minimal domination sets of a graph. It is known that the corresponding decision problem belongs to coNP, but still open whether there exists an output-polynomial-time algorithm.

4 Partial transversal hypergraph enumeration

We introduce the partial (or incomplete) search algorithm for enumerating minimal transversals of an hypergraph $\mathcal{H}$. The search space is the set of all transversals which is very large. The strategy is divided into two steps:

- The initialization procedure considers a transversal T of $\mathcal{H}$ and then applies a reduction (at random) algorithm to T in order to obtain a minimal transversal T_m of $\mathcal{H}$. This step is detailed in section 4.1.
- The local search algorithm considers a minimal transversal T_m and then applies local changes to T_m in which we add and delete vertices according to some ordering of the vertices. This step is detailed in section 4.2

These steps are repeated for at most k transversals depending on the input hypergraph $\mathcal{H}$. Figure 1 illustrates the proposed approach.

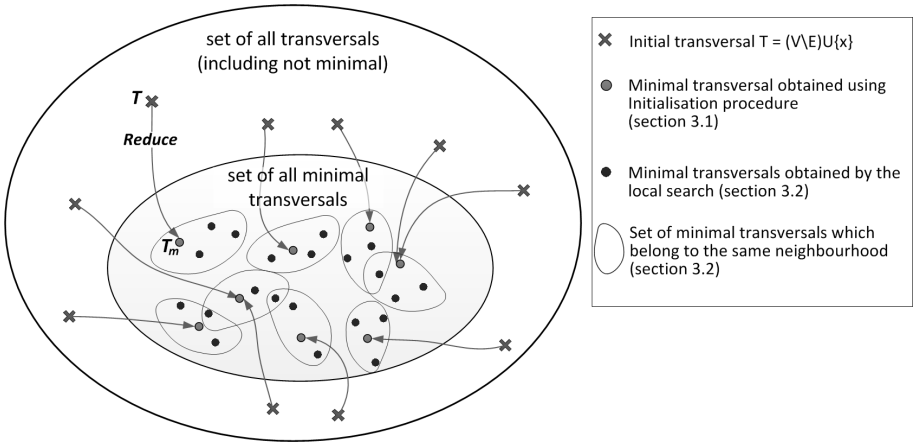


Fig. 1. Approach to partial enumeration of minimal transversals

4.1 Initialization

Let $\mathcal{H}(V, \mathcal{E})$ be the input hypergraph, $E \in \mathcal{E}$ and $x \in E$. The initialization step starts with the transversal $(V \setminus E) \cup \{x\}$ and then applies a reduction

algorithm to obtain a minimal transversal. The following property shows that the set $(V \setminus E) \cup \{x\}$ is a transversal and any minimal transversal contained in $(V \setminus E) \cup \{x\}$ contains the vertex x.

Property 1. Let $\mathcal{H} = (V, \mathcal{E})$ be a simple hypergraph, $E \in \mathcal{E}$ and $x \in E$. Then $(V \setminus E) \cup \{x\}$ a transversal of $\mathcal{H}$. Moreover, any minimal transversal $T \subseteq (V \setminus E) \cup \{x\}$ will contain x.

Proof. Let $\mathcal{H} = (V, \mathcal{E})$ be a simple hypergraph and $E' \in \mathcal{E}$ with $E \neq E'$. Since $\mathcal{H}$ is simple then there exists at least one element $y \in E'$ such that $y \notin E$. So $y \in (V \setminus E)$ and thus $E' \cap (V \setminus E) \neq \emptyset$. We conclude that $(V \setminus E) \cup \{x\}$ is a transversal since $x \in E$.

Now let $T \subseteq (V \setminus E) \cup \{x\}$ be a minimal transversal. Then $E \cap (V \setminus E) \cup \{x\} = \{x\}$ and thus x must belong to T otherwise T does not intersect E. □

According to property 1, we can apply the initialization procedure to any pair (x, E) where $E \in \mathcal{E}$ and $x \in E$. In other words, the initialization is applied to at most k transversals of $\mathcal{H}$ as shown in Algorithm 1.

Algorithm 1: Initialization

Input : A hypergraph $\mathcal{H}(V, \mathcal{E})$ and σ an ordering of V
Output: A sample of minimal transversals
begin
  $STRANS = \emptyset$;
  for $E \in \mathcal{E}$ **do**
    for $x \in E$ **do**
      $T = (V \setminus E) \cup \{x\}$;{Initial transversal}
      $T_m = Reduce(T, \sigma)$;
      $STRANS = STRANS \cup \{T_m\}$;
  return($STRANS$);

Now we describe the reduction process, which takes a transversal T and a random ordering σ of V and returns a minimal transversal T_m. Indeed, we delete vertices from T according to the ordering σ until we obtain a minimal transversal.

Algorithm 2: Reduce(T, σ)

Input : A transversal T and an ordering $\sigma = \sigma_1 ... \sigma_{|V|}$ of the vertices of $\mathcal{H}$.
Output: A minimal transversal

for $i = 1$ *to* $|V|$ **do**
  if $T \setminus \{\sigma_i\}$ *is a transversal* **then**
    $T \leftarrow T \setminus \{\sigma_i\}$;
Return(T);

Example 2 (continued). Suppose we are given the hypergraph in example 1 and $\sigma = (1, 2, 3, 4, 5)$ for the input to Algorithm 1. First, it takes the hyperedge $E = \{1, 3, 4\}$ and for $x = 1$ we obtain the minimal transversal $\{1\}$, for $x = 3$ we obtain $\{2, 3\}$ and for $x = 4$ we obtain $\{2, 4, 5\}$. Then the algorithm continue with the hyper edges $\{1, 3, 5\}$ and $\{1, 2\}$. Finally the algorithm returns $STRANS = \{\{1\}, \{2, 3\}, \{2, 4, 5\}\}$, i.e. the other iterations do not add new minimal transversals.

Theorem 1. *Algorithm 1 computes at most k minimal transversals of an input hypergraph* $\mathcal{H} = (V, \mathcal{E})$.

Proof. The initialization procedure considers at most k minimal transversals of $\mathcal{H}$. Since a minimal transversal can be obtained several times, the result follows. □

The following proposition shows that any minimal transversal of the hypergraph $\mathcal{H} = (V, \mathcal{E})$ can be obtained using the initialization procedure. Indeed, the choice of the ordering σ is important in the proposed strategy.

Proposition 1. *Let* $\mathcal{H} = (V, \mathcal{E})$ *be an hypergraph and* T *be a minimal transversal of* $\mathcal{H}$. *Then, there exists a total order* σ, $E \in \mathcal{E}$ *and* $x \in E$ *such that* $T = Reduce((V \setminus E) \cup \{x\}, \sigma)$.

Proof. Let T be a minimal transversal of $\mathcal{H} = (V, \mathcal{E})$. Then there exists at least one hyperedge $E \in \mathcal{E}$ such that $T \cap E = \{x\}$, $x \in V$, otherwise T is not minimal. Thus $T \subseteq (V \setminus E) \cup \{x\}$. Now, if we take the elements that are not in T before the elements in T in σ, the algorithm $Reduce((V \setminus E) \cup \{x\}, \sigma)$ returns T. □

The initialization procedure guaranties that for any vertex $x \in V$ at least one minimal transversal containing x is listed. The experiments in section 5, shows the sample of minimal transversals obtained by the initialization procedure is a representative sample of the set of all minimal transversals.

4.2 Local search algorithms

The local search algorithm takes each minimal transversal found in the initialization step and searches for new minimal transversals to improve the initial solution. The search of neighbors is based on vertices orderings.

Let $\mathcal{H} = (V, \mathcal{E})$ be an hypergraph and $x \in V$. We define the frequency of x as the number of minimal transversals of $\mathcal{H}$ that contain x. The algorithm takes a minimal transversal T and a bound $Nmax$ which bounds the number of iterations and the number of neighboors generated by T. Each iteration of the while loop, starts with a minimal transversal T and computes two orderings as follows:

- σ^c is an ordering according to the increasing order of frequency of vertices in $V \setminus T$ in minimal transversals already obtained by the current call. This ordering has a better coverage of the solution set, i.e. by keeping the rarest vertices in the transversals.

– σ is a random ordering of the vertices in T.

Algorithm 3: Neighboor$(T, Nmax)$

Input : A minimal transversal T of $\mathcal{H} = (V, \mathcal{E})$ and an integer $Nmax$
Output: A set of minimal transversals

$Q = T$;
$i = 1$;
while $i \leq Nmax$ **do**
  $\sigma^c \leftarrow$ the set $V \setminus T$ sorted in increasing order of frequency of vertices in minimal transversals in Q;
  $\sigma \leftarrow$ sort T at random;
  Add elements the elements in σ^c to T until a vertex $x \in T \setminus \sigma^c$ becomes redundant in T;
  $T =$Reduce(T, σ);
  $Q = Q \cup \{T\}$;
  $i = i + 1$;
return(Q);

Now we give the global procedure of the proposed approach.

Algorithm 4: Global procedure for partial enumeration of minimal transversals

Input : An hypergraph $\mathcal{H} = (V, \mathcal{E})$ and an integer $Nmax$
Output: A sample of minimal transversals of $\mathcal{H}$

$\sigma =$choose a random ordering of V;
$STRAN = Q = Initialization(\mathcal{H}, \sigma)$;
while $Q \neq \emptyset$ **do**
  $T =$ choose a minimal transversal T in Q;
  $STRANS = STRANS \cup Neighboor(T, Nmax)$;
Return$(STRANS)$;

In the following, we describe experiments to evaluate the results that have been obtained.

5 Experimentation

The purpose of the experiments is to see if the proposed approach allow us to generate a *representative* set of solutions. For this reason, we have conducted the experiments on two different classes of hypergraphs (see [20]) for which the number of minimal transversals is huge compared to the size of the input. We use Uno's Algorithm SHD (Sparsity-based Hypergraph Dualization, ver. 3.1) [20], to enumerate all minimal transversals. The experiments are done using linux CentOS cpu Intel Xeon 3.6 GHz and C++ language.

In the following, we denote $\mathcal{T}_{partial}$ the set of minimal transversals generated by Algorithm 4, and $\mathcal{T}_{exact}$ the set of all minimal transversals. First, we analyze

the percentage $\frac{\mathcal{T}_{partial}}{\mathcal{T}_{exact}}$ and then we evaluate the representativeness of the sample $\mathcal{T}_{partial}$.

5.1 The size of $\mathcal{T}_{partial}$

We will distinguish between minimal transversals that are obtained using Algorithm 1 (or the initialization procedure) and those that are generated using the local search. For these tests we set the maximal number of neighboors $Nmax$ to 3.
Tables 1 and 2 show the results for the two classes of hypergraph instances, namely "lose" and random "p8".

The first three columns have the following meaning:

- *instance*: instance name.
- *instance size*: the size of the instance (number of edges × number of vertices).
- *total # of transv.*: the exact number of minimal transversals $\mid \mathcal{T}_{exact} \mid$.

The second (resp. last) three columns give the results for the initialization procedure (resp. Global algorithm):

- *# transv. found*: the number of minimal transversals found.
- *% transv. found*: the percentage of minimal transversals found.
- *cpu (s)*: the run time in seconds

			Initialisation algorithm			enumeration algorithm with local search		
instance	instance size	total # of transv.	# transv. found	% transv. found	cpu (s)	# transv. found	% transv. found	cpu (s)
lose100.dat	100 * 76	2 341	529	22.6	<0.1	1 256	53.7	<0.1
lose200.dat	200 * 76	22760	806	3.5	<0.1	2 621	11.5	0.1
lose400.dat	400 * 78	33 087	1 374	4.2	<0.1	4 508	13.6	0.3
lose800.dat	800 * 80	79 632	3 142	4.0	<0.1	10 252	12.9	1.2
lose1600.dat	1 600 * 80	212 761	7 475	3.5	0.3	23 929	11.3	4.8
lose3200.dat	3 200 * 80	2 396 735	19 193	0.8	1.6	66 269	2.8	26.3
lose6400.dat	6 400 * 80	4 707 877	36 295	0.8	9.0	136 109	2.9	349.0
lose12800.dat	12 800 * 84	16 405 082	81 946	0.5	38.7	295 847	1.8	1 146.8
lose.dat	16 635 * 84	39 180 611	117 417	0.3	67.4	427 417	1.1	1 524.1

Table 1. Results for all "*lose*" instances

According to these tests, we can see that the percentage of minimal transversals found using the initialization procedure is very low, but it decreases as far as the size of $\mathcal{T}_{exact}$ increases. Clearly, this percentage is strongly related to the input. Indeed, the number k (entropy) of the hypergraph increases according to the size of the input hypergraph. We can also see that the local search increases significantly the number of solutions found by a factor 2 to 2.5 approximatively.

			Initialisation algorithm			enumeration algorithm with local search		
instance	instance size	total # of transv.	# transv. found	% transv. found	cpu (s)	# transv. found	% transv. found	cpu (s)
p8_200.dat	200 * 50	52 395	5 790	11.1	0.1	16 339	31.2	0.3
p8_600.dat	600 * 50	170 418	19 581	11.5	0.5	56 893	33.4	2.1
p8_1000.dat	1 000 * 50	364 902	31 007	8.5	1.4	82 619	22.6	5.5
p8_2000.dat	2 000 * 50	795 378	62 618	7.9	5.2	206 793	26.0	20.3
p8_4000.dat	4 000 * 50	1 320 830	131 422	10.0	21.5	390 231	29.5	88.2
p8_8000.dat	8 000 * 50	4 201 736	240 913	5.7	103.4	703 595	16.8	399.3
p8_10000.dat	10 000 * 50	5 378 267	306 922	5.7	168.1	1 002 561	18.6	489.2
p8_16000.dat	16 000 * 50	6 537 313	511 639	7.8	446.3	1 716 569	26.3	1 383.4
p8_20000.dat	20 000 * 50	7 628 650	632 227	8.3	685.9	2 007 667	26.3	2 042.7
p8_32000.dat	32 000 * 50	16 344 095	938 177	5.7	1 730.3	2 590 215	15.9	6 300.0
p8_64000.dat	64 000 * 50	35 072 428	1 845 733	5.3	6 955.1	6 266 167	17.9	17 533.0

Table 2. Results for all "*p8*" instances

But it remains coherent with the chosen value $Nmax = 3$. It argues that the local search finds other transversals that are not found either by the initialization procedure nor previous local search. Notice that the parameter $Nmax$ can be increased whenever the size of $\mathcal{T}_{partial}$ is not sufficient.

5.2 The representativeness of $\mathcal{T}_{partial}$

To evaluate the representativeness of $\mathcal{T}_{partial}$, we consider three criteria:

- Size of the minimal transversals in $\mathcal{T}_{partial}$.
- Frequency of vertices in $\mathcal{T}_{partial}$.
- Lexicographical rank of the minimal transversals in $\mathcal{T}_{partial}$.

Each criteria is illustrated using a bar graph for two instances from different classes. The bar graphs in figures 2 and 3 are surprising. Indeed the bar graphs vary nearly in the same manner with respect to the initialization and the local search algorithm for all the considered criteria.

Figures 2(a) 3(a) show that the percentage of minimal transversals of each size (found either by the initialization procedure and local search) fits the percentage of all minimal transversals that are found.

Figures 2(b) and 3(b) show that the same analysis holds when ordering minimal transversals lexicographically (e.g. based on a total ordering of vertices). Clearly, the lexicographical rank of a minimal transversal belongs to the interval $[1..2^{|V|}]$. For visualization aspect, we divide this interval into $|V|$ subintervals, where the subinterval i contains the number of minimal transversals with a rank $r \in [\frac{2^{|V|}}{|V|}i; \frac{2^{|V|}}{|V|}(i+1)[$, $(i = 0, \ldots, |V| - 1)$.

Figures 2(c) and 3(c) confirm this behavior when considering frequency of vertices. Indeed, frequency of vertices in minimal transversals in $\mathcal{T}_{partial}$ is the same when considering all minimal transversals, i.e.. the set $\mathcal{T}_{exact}$.

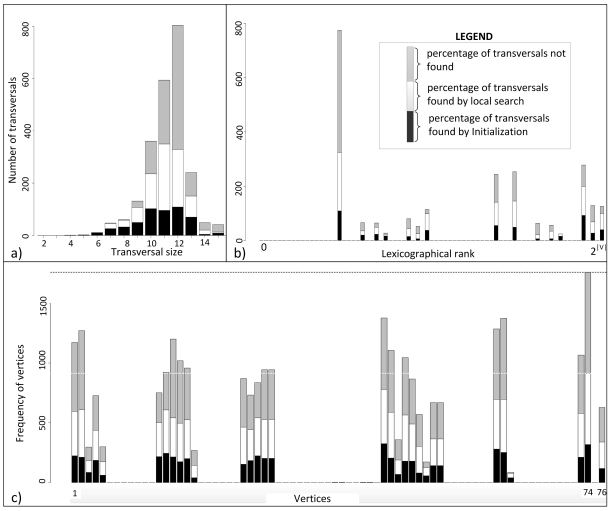


Fig. 2. The bar graph for "lose100": a) number of minimal transversals per size b) number of minimal transversals according the lexicographical rank; c) Frequency of vertices in minimal transversals.

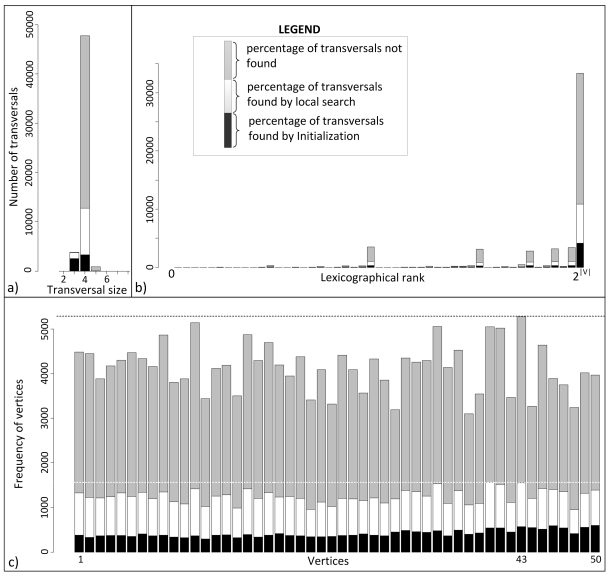


Fig. 3. The bar graph for "p8_200"; a) number of minimal transversals per size; b) number of minimal transversals according the lexicographical rank; c) Frequency of vertices in minimal transversals.

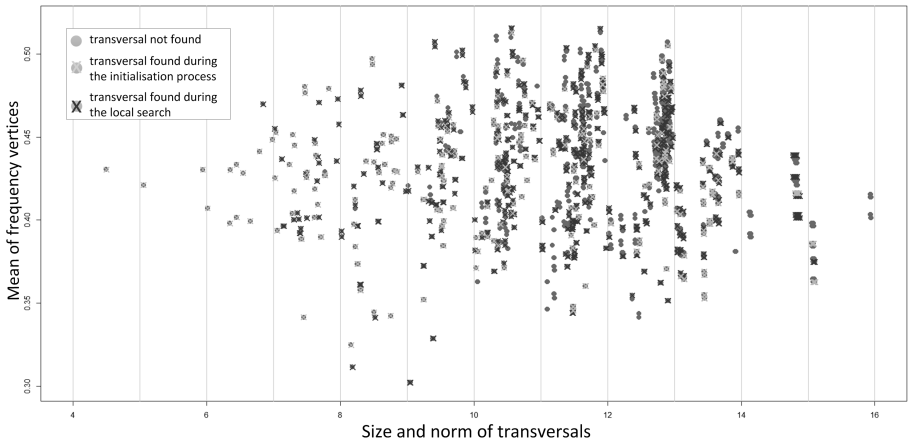


Fig. 4. Visualizing the solutions space of "lose100". The abscissa is given by the size of the transversal (transversals of the same size are spread out using a norm) and the ordinate corresponds to the frequency of its vertices.

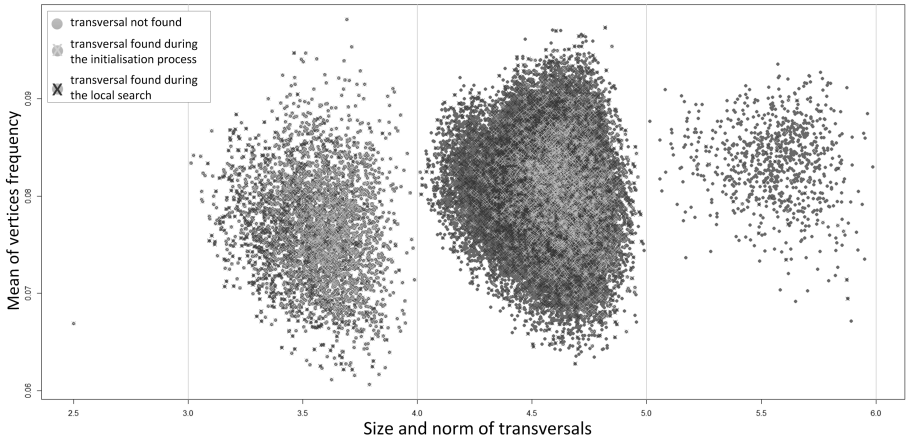


Fig. 5. Visualizing the solutions space of "p8_200". The abscissa is given by the size of the transversal (transversals of the same size are spread out using a norm) and the ordinate corresponds to the frequency of its vertices.

Figures 4 and 5 show that the set $\mathcal{T}_{partial}$ is also representative even when considering minimal transversals with the same size. Indeed, minimal transversals having the same size are spread out using a norm. We notice that the points corresponding to minimal transversals in $\mathcal{T}_{partial}$ are scattered in the image.

This experiment allows us to conclude that the sample $\mathcal{T}_{partial}$ produced by Algorithm 4 is representative relatively to the criteria under consideration. Other results can be found in http://www2.isima.fr/~toussain/

6 Conclusion and discussions

We are convinced that the initialization procedure is the most important in this approach. Indeed, the set of minimal transversals obtained using this procedure is a representative sample, since it garantee that for any vertex of the hypergraph there is at least one minimal transversal which contains it (see property 1). Moreover the local search procedure can be used to increase the number of solutions, and as we have seen in the experiments, it keeps the same properties as the initialization procedure.

We hope that this approach improves enumeration in big data and will be of interests to the readers to investigate heuristics methods [6] for enumeration problems.

This paper opens new challenges related to partial and approximate enumeration problems. For example, given an hypergraph $\mathcal{H} = (V, \mathcal{E})$, is there an algorithm that for any given ε, it enumerates a set $\mathcal{T}_{partial} \subseteq Tr(\mathcal{H})$ such that $(1 - \varepsilon)|Tr(\mathcal{H})| \leq |\mathcal{T}_{partial}| \leq |Tr(\mathcal{H})|$? We also require that the algorithm is output-polynomial in the sizes of $\mathcal{H}$, $\mathcal{T}_{partial}$ and $\frac{1}{\varepsilon}$. To our knowledge, there is no work on approximate algorithms for enumeration problems, but results on approximate counting problems may be applied [16].

Acknowledgment: This work has been funded by the french national research agency (ANR DAG project, 2009-2013) and CNRS (Mastodons PETASKY project, 2012-2015).

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *ACM SIGMOD 1993, Washington D.C.*, pages 207–216, 1993.
2. J. Y. Chen and S. Lonardi. *Biological Data Mining.* Chapman and Hall/CRC, 2009.
3. T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.
4. T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. *SIAM J. Comput.*, 32(2):514–537, 2003.
5. T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008.

6. T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.
7. M. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21:618–628, 1996.
8. L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), Sept. 2006.
9. P. Golovach, P. Heggernes, D. Kratsch, and Y. Villanger. An incremental polynomial time algorithm to enumerate all minimal edge dominating sets. *Algorithmica*, 72(3):836–859, 2015.
10. D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharm. Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2):140–174, 2003.
11. D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *PODS*, pages 209–216, 1997.
12. M. Jelassi, C. Largeron, and S. Ben Yahia. Concise representation of hypergraph minimal transversals: Approach and application on the dependency inference problem. In *Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on*, pages 434–444, May 2015.
13. D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.
14. M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. On the enumeration of minimal dominating sets and related notions. *SIAM Journal on Discrete Mathematics*, 28(4):1916–1929, 2014.
15. L. Khachiyan, E. Boros, K. M. Elbassioni, and V. Gurvich. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discrete Applied Mathematics*, 154(16):2350–2372, 2006.
16. J. Liu and P. Lu. Fptas for counting monotone cnf. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 1531–1548. SIAM, 2015.
17. K. Murakami and T. Uno. Efficient algorithms for dualizing large-scale hypergraphs. *Discrete Applied Mathematics*, 170:83–94, 2014.
18. L. Nourine and J.-M. Petit. Extending set-based dualization: Application to pattern mining. In *ECAI*, pages IOS Press ed, Montpellier, France, 2012.
19. O. Raynaud, R. Medina, and C. Noyer. Twin vertices in hypergraphs. *Electronic Notes in Discrete Mathematics*, 27:87–89, 2006.
20. T. Uno. http://research.nii.ac.jp/ uno/dualization.html.
21. C. A. Weber, J. R. Current, and W. Benton. Vendor selection criteria and methods. *European Journal of Operational Research*, 50(1):2 – 18, 1991.

An Introduction to Semiotic-Conceptual Analysis with Formal Concept Analysis

Uta Priss

Zentrum für erfolgreiches Lehren und Lernen
Ostfalia University of Applied Sciences
Wolfenbüttel, Germany
www.upriss.org.uk

Abstract. This paper presents a formalisation of Peirce's notion of 'sign' using a triadic relation with a functional dependency. The three sign components are then modelled as concepts in lattices which are connected via a semiotic mapping. We call the study of relationships relating to semiotic systems modelled in this manner a *semiotic-conceptual analysis*. It is argued that semiotic-conceptual analyses allow for a variety of applications and serve as a unifying framework for a number of previously presented applications of FCA.

1 Introduction

The American philosopher C. S. Peirce was a major contributor to many fields with a particular interest in semiotics. The following quote shows one of his definitions for the relationships involved in using a sign:

> A REPRESENTAMEN is a subject of a triadic relation TO a second, called its OBJECT, FOR a third, called its INTERPRETANT, this triadic relation being such that the REPRESENTAMEN determines its interpretant to stand in the same triadic relation to the same object for some interpretant. (Peirce, CP 1.541)[1]

According to Peirce a sign consists of a physical form (representamen) which could, for example, be written, spoken or represented by neurons firing in a brain, a meaning (object) and another sign (interpretant) which mirrors the original sign, for example, in the mind of a person producing or observing a sign. It should be pointed out that the use of the term 'relation' by Peirce is not necessarily the same as in modern mathematics which distinguishes more clearly between a 'relation' and its 'instances'. Initially Peirce even referred to mathematical relations as 'relatives' (Maddux, 1991).

We have previously presented an attempt at mathematically formalising Peirce's definition (Priss, 2004). In our previous attempt we tried to presuppose as few assumptions about semiotic relations as possible which led to a fairly open structural description which, however, appeared to be limited with respect to usefulness in applications.

[1] It is customary among Peirce scholars to cite Peirce in this manner using an abbreviation of the publication series, volume number and paragraph or page numbers.

 Published in Sadok Ben Yahia, Jan Konecny (Eds.): CLA 2015, pp. 135–146, ISBN 978–2–9544948–0–7, Blaise Pascal University, LIMOS laboratory, Clermont-Ferrand, 2015.

Now we are presenting another formalisation which imposes a functional dependency on the triadic relation. The notions from Priss (2004) are translated into this new formalism which is in many ways simpler, more clearly defined and appears to be more useful for applications.

In order to avoid confusion with the notion of 'object' in FCA[2], we use the term 'denotation' instead of Peirce's 'object' in the remainder of this paper. We translate the first half of Peirce's definition into modern language as follows: "A representamen is a parameter of a function resulting in a second, called its denotation, where the third, the function instance, is called interpretant." – or in other words as a function of type 'third(first) = second'. In our modelling, a set of such functions together with their parameter/value pairs constitute a triadic semiotic relation. We use Peirce's notion of 'interpretant' for the function instances whereas the functions themselves are called 'interpretations'. A sign is then an instance of this triadic relation consisting of representamen, denotation and interpretation. This is more formally defined in the next section.

The second half of Peirce's definition refers to the mental image that a sign invokes in participants of communication acts. For Peirce, interpretants are mental images which can themselves be thought about and thus become representamens for other interpretants and so on. Because the focus of this paper is on formal, not natural languages, mental images are not important. We suggest that in formal languages, interpretants are not mental representations but instead other formal structures, for example, states in a computer program.

The data structures used in formal languages (such as programming languages, XML or UML) can contain a significant amount of complexity. A semiotic-conceptual analysis as proposed in this paper allows to investigate the components of such structures as signs with their representamens, denotations and interpretations and their relationships to each other. As a means of structuring the semiotic components we use FCA concept lattices.

It should be noted that there has recently been an increase of interest in triadic FCA (e.g., Gnatyshak et al. (2013), Belohlavek & Osicka (2012)) which could also be used to investigate triadic semiotic relations. But Peirce tends to see triadic relations as consisting of three components of increasing complexity:

> The First is that whose being is simply in itself, not referring to anything nor lying behind anything. The Second is that which is what it is by force of something to which it is second. The Third is that which is what it is owing to things between which it mediates and which it brings into relation to each other. (Peirce, EP 1:248; CP 1.356)

In our opinion this is better expressed by a function instance of type 'third(first) = second' than by an instance '(first, second, third)' of a triadic relation. Other researchers have already suggested formalisations of Peirce's philosophy. Interestingly, Marty (1992), Goguen (1999) and Zalamea (2010) all suggest using Category Theory

[2] Because Formal Concept Analysis (FCA) is the main topic of this conference, this paper does not provide an introduction to FCA. Information about FCA can be found, for example, on-line (http://www.fcahome.org.uk) and in the main FCA textbook by Ganter & Wille (1999).

for modelling Peirce's philosophy even though they appear to have worked independently of each other. Marty even connects Category Theory with FCA in his modelling. Goguen develops what he calls 'algebraic semiotics'. Zalamea is more focussed on Existential Graphs than semiotics (and unfortunately most of his papers are in Spanish). Nevertheless our formalisation is by far not as abstract as any of these existing formalisations which are therefore not further discussed in this paper.

This paper has five further sections. Section 2 presents the definitions of signs, semiotic relations and NULL-elements. Section 3 continues with defining concept lattices for semiotic relations. Section 4 explains degrees of equality among signs. Section 5 discusses mappings among the lattices from Section 3 and presents further examples. The paper ends with a concluding section.

2 Core definitions of a semiotic-conceptual analysis

The main purpose of this work is to extend Peirce's sign notion to formal languages such as computer programming languages and formal representations. Figure 1 displays a simple Python program which is called 'Example 1' in the remainder of this paper. The table underneath shows the values of the variables of Example 1 after an execution. The variables are representamens and their values are denotations. Because Peirce's definition of signs seems to indicate that there is a separate interpretant for each sign, there are at least eight different interpretants in column 3 of the table. It seems more interesting, however, to group interpretants than to consider them individually. We call such groupings of interpretants *interpretations*. In natural language examples, one could group all the interpretants that belong to a sentence or paragraph. In programming languages starting a loop or calling a subroutine might start a new interpretation. As a condition for interpretations we propose that each representamen must have a unique denotation in an interpretation, or in other words, interpretations are functions. There are many different possibilities for choosing sets of interpretations. Two possibilities, called I_A and I_B in the rest of the paper, are shown in the last two columns of the table. Each contains two elements which is in this case the minimum required number because some variables in Example 1 have two different values. In our modelling an interpretant corresponds to a pair of representamen and interpretation. For R and I_A there are ten interpretants (and therefore ten signs) whereas there are eight for R and I_B. The first three columns of the table can be automatically derived using a debugging tool. The interpretants are numbered in the sequence in which they are printed by the debugger.

Definition 1: A *semiotic relation* $\overline{S} \subseteq I \times R \times D$ is a relation between three sets (a set R of *representamens*, a set D of *denotations* and a set I of *interpretations*) with the condition that any $i \in I$ is a partial function $i : R \nrightarrow D$. A relation instance (i, r, d) with $i(r) = d$ is called a *sign*. In addition to $\overline{S}$, we define the *semiotic (partial) mapping* $S : I \times R \nrightarrow D$ with $S(i, r) = d$ iff $i(r) = d$. The pairs (i, r) for which d exists are called *interpretants*.

It follows that there are as many signs as there are interpretants. Example 1 shows two semiotic relations using either I_A or I_B for the interpretations. The interpretations firstLoop, secondLoop and firstValue are total functions. The interpretation second-

```
input_end = "no"
while input_end != "yes":
    input1 = raw_input("Please type something: ")
    input2 = raw_input("Please type something else: ")
    if (input1 == input2):
       counter = 1
       error = 1
       print "The two inputs should be different!"
    else:
       counter = 2
    input_end = raw_input("End this program? ")
```

representamens R (variables)	denotations D (values)	interpretants	interpretations I_A ($\sim$10 interpretants)	interpretations I_B ($\sim$ 8 interpretants)
input1	"Hello World"	j1	firstLoop	firstValue
input2	"Hello World"	j2	firstLoop	firstValue
counter	1	j3	firstLoop	firstValue
input_end	no	j4	firstLoop	firstValue
error	1	j5	firstLoop	firstValue
input1	"Hello World"	j6 (or j1)	secondLoop	firstValue
input2	"How are you"	j7	secondLoop	secondValue
counter	2	j8	secondLoop	secondValue
input_end	yes	j9	secondLoop	secondValue
error	1	j10 (or j5)	secondLoop	firstValue

Fig. 1. A Python program (called 'Example 1' in this paper)

Value is a partial function. Because for (i_1, r_1, d_1) and (i_2, r_2, d_2), $i_1 = i_2, r_1 = r_2 \Rightarrow d_1 = d_2$, it follows that all $r \in R$ with $r(i) = d$ are also partial functions $r : I \nrightarrow D$. The reason for having a relation $\overline{S}$ and a mapping S is because Peirce defines a relation but in applications a mapping might be more usable. In this paper the sets R, D and I are meant to be finite and not in any sense universal but built for an application. The assignment operation (written ':=' in mathematics or '=' in programming languages) is an example of $i(r) = d$ except that i is usually implied and not explicitly stated in that case.

Using the terminology from database theory, we call a semiotic relation *a triadic relation with functional dependency*. This is because, on the one hand, Peirce calls it not a mapping but a 'triadic relation', on the other hand, without this functional dependency it would not be possible to determine the meaning of a sign given its representamen and an interpretation. Some philosophers might object to Definition 1 because of the functional dependency. We argue that the added functional dependency yields an interesting structure which can be explored as shown in this paper.

The idea of using interpretations as a means of assigning meaning to symbols is already known from formal semantics and model theory. But this paper has a different focus by treating interpretations and representamens as dual structures. Furthermore in applications, $S(i, r)$ might be implemented as an algorithmic procedure which determines d for r based on information about i at runtime. A debugger as in Example 1

is not part of the original code but at a meta-level. Since the original code might request user input (as in Example 1), the relation instances (i, r, d) are only known while or after the code was executed. Thus the semiotic relation is dynamically generated in an application. This is in accordance with Peirce's ideas about how it is important for semiotics to consider how a sign is actually *used*. The mathematical modelling (as in Definition 1) which is conducted after a computer program finished running, ignores this and simply considers the semiotic relation to be statically presented.

Priss (2004) distinguishes between triadic signs and anonymous signs which are less complex. In the case of anonymous signs, the underlying semiotic relation can be reduced to a binary or unary relation because of additional constraints. Examples of anonymous signs are constants in programming languages and many variables used in mathematical expressions. For instance, the values of variables in the Pythagorean equation $a^2 + b^2 = c^2$ are all the values of all possibly existing right-angled triangles. But, on the one hand, if $a^2 + b^2 = c^2$ is used in a proof, it is fine to assume $|I| = 1$ because within the proof the variables do not change their values. On the other hand, if someone uses the formula for an existing triangle, one can assume $S(i, r) = r$ because in that case the denotations can be set equal to the representamens. Thus within a proof or within a mathematical calculation variables can be instances of binary or unary relations and thus anonymous signs. However, in the following Python program:

```
a = input("First side: ")
b = input("Second side: ")
print a*a + b*b
```

the values of the variables change depending on what is entered by a user. Here the signs a and b are triadic.

A sign is usually represented by its representamen. In a semiotic analysis it may be important to distinguish between 'sign' and 'representamen'. In natural language this is sometimes indicated by using quotes (e.g., the word 'word'). In Example 1, the variable 'input1' is a representamen whereas the variable 'input1' with a value 'Hello World' in the context of firstLoop is a sign. It can happen that a representamen is taken out of its context of use and loses its connection to an interpretation and a denotation. For example, one can encounter an old file which can no longer be read by any current program. But a sign always has three components (i, r, d). Thus just looking at the source code of a file creates an interpretant in that person's mind even though this new sign and the original sign may have nothing in common other than the representamen. Using the next definition, interpretations that are partial functions can be converted into total functions.

Definition 2: For a semiotic relation, a *NULL-element* $d_\perp$ is a special kind of denotation with the following conditions: (i) $i(r)$ undefined in $D \Rightarrow i(r) := d_\perp$ in $D \cup \{d_\perp\}$. (ii) $d_\perp \in D \Rightarrow$ all i are total functions.

Thus by enlarging D with one more element, one can convert all i into total functions. If all i are already total functions, then $d_\perp$ need not exist. The semiotic mapping S can be extended to a total function $S : I \times R \to D \cup \{d_\perp\}$. There can be different reasons for NULL-elements: caused by the selection of interpretations or by the code itself. Variables are successively added to a program and thus undefined for any interpretation that occurs before a variable is first defined. In Example 1, secondValue is a partial function because secondValue(input1) = secondValue(error) = $d_\perp$. But I_A shows

that all interpretations can be total functions. On the other hand, if a user always enters two different values, then the variable 'error' is undefined for all interpretations. This could be avoided by changing the code of Example 1. In more complex programs it may be more difficult to avoid $d_\perp$, for example if a call to an external routine returns an undefined value. Programming languages tend to allow operations with $d_\perp$, such as evaluating whether a variable is equal to $d_\perp$, in order to avoid run-time errors resulting from undefined values. Because the modelling in the next section would be more complex if conditions for $d_\perp$ were added we decided to mostly ignore $d_\perp$ in the remainder of this introductory paper.

3 Concept lattices of a semiotic relation

In order to explore relationships among signs we are suggesting to model the components of signs as concept lattices. The interpretations which are (partial) functions from R to D then give rise to mappings between the lattice for R and the lattice for D. Figure 2 shows an example of concept lattices for the semiotic relation from Example 1. The objects are the representamens, denotations and interpretations of Example 1. The attributes are selected for characterising the sets and depend on the purpose of an application. If the denotations are values of a programming language, then data types are a fairly natural choice for the attributes of a denotation lattice.

Attributes for representamens should focus on representational aspects. In Example 1, all input variables start with the letters 'input' because of a naming style used by the programmer of that program. In some languages certain variables start with upper or lowercase letters, use additional symbols (such as '@' for arrays) or are complex structures (such as 'root.find("file").attrib["size"]') which can be analysed in a representamen lattice. In strongly-typed languages, data types could be attributes of representamens but in languages where variables can change their type, data types do not belong into a representamen lattice. Rules for representamens also determine what is to be ignored. For example white space is ignored in many locations of a computer program. The font of written signs is often ignored but mathematicians might use Latin, Greek and Fraktur fonts for representamens of different types of denotations.

One way of deriving a lattice for interpretations is to create a partially ordered set using the ordering relation as to whether one interpretation precedes another one or whether they exist in parallel. A lattice is then generated using the Dedekind closure. In Figure 2 the attributes represent some scaling of the time points of the interpretations. Thus temporal sequences can be expressed but any other ordering can be used as well.

Definition 3: For a set R of representamens, a concept lattice $B(R, M_R, J_R)$ is defined where M_R is a set of attributes used to characterise representamens and J_R is a binary relation $J_R \subseteq R \times M_R$. $B(R, M_R, J_R)$ is called *representamen lattice*. $B(R, M_R, J_R)$ is *complete for a set of interpretations*[3] if for all $r \in R$: $\forall_{i \in I} : \gamma(r_1) = \gamma(r_2) \Rightarrow i(r_1) = i(r_2)$ and $\gamma(r_1) \neq \gamma(r_2) \Rightarrow \exists_{i \in I} : i(r_1) \neq i(r_2)$.

Definition 4: For a set I of interpretations, a concept lattice $B(I, M_I, J_I)$ is defined where M_I is a set of attributes used to characterise the interpretations and J_I is a binary

[3] For an object o its object concept $\gamma(o)$ is the smallest concept which has the object in its extension.

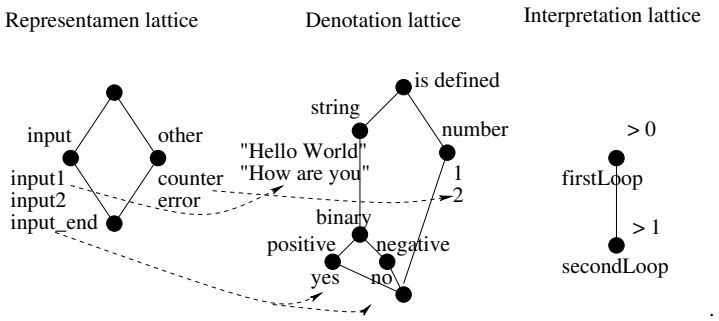


Fig. 2. Lattices for Example 1

relation $J_I \subseteq I \times M_I$. $B(I, M_I, J_I)$ is called *interpretation lattice*. $B(I, M_I, J_I)$ is *complete for a set of representamens* if for all $i \in I$: $\forall_{r \in R} : \gamma(i_1) = \gamma(i_2) \Rightarrow i_1(r) = i_2(r)$ and $\gamma(i_1) \neq \gamma(i_2) \Rightarrow \exists_{r \in R} : i_1(r) \neq i_2(r)$.

The representamen lattice in Figure 2 is not complete for I_A because, for example, 'input_end' and 'input1' have different denotations. The interpretation lattice is complete for R because no objects have the same object concept and firstLoop and secondLoop have different denotations, for example, for 'counter'. Completeness means that exactly those representamens or interpretations that share their object concepts can be used interchangeably without any impact on their relationship with the other sets. The dashed lines in Figure 2 are explained below in Section 5.

Definition 5: For a set $D \setminus \{d_\perp\}$ of denotations, a concept lattice $B(D, M_D, J_D)$ is defined where M_D is a set of attributes used to characterise the denotations and J_D is a binary relation $J_D \subseteq D \times M_D$. $B(D, M_D, J_D)$ is called *denotation lattice*.

4 Equality and other sign properties

Before continuing with the consequences of the definitions of the previous section, equality of signs should be discussed because there are different degrees of equality. Two signs, (i_1, r_1, d_1) and (i_2, r_2, d_2), are equal if all three components are equal. Because of the functional dependency this means that two signs are equal if $i_1 = i_2$ and $r_1 = r_2$. In normal mathematics the equal sign is used for denotational equality. For example, $x = 5$ means that x has the value of 5 although clearly the representamen x has nothing in common with the representamen 5. Since signs are usually represented by their representamens denotational equality needs to be distinguished from equality between signs. Denotational equality is called 'strong synonymy' in the definition below. Even strong synonymy is sometimes too much. For example natural language synonyms (such as 'car' and 'automobile') tend to always still have subtle differences in meaning. In programming languages, if a counter variable increases its value by 1, it is still thought of as the same variable. But if such a variable changes from '3' to 'Hello World' and then to '4', depending on the circumstances, it might indicate an

error. Therefore we are defining a tolerance relation[4] $T \subseteq D \times D$ to express that some denotations are close to each other in meaning. With respect to the denotation lattice, the relation T can be defined as the equivalence relation of having the same object concept or via a distance metric between concepts. The following definition is an adaptation from Priss (2004) that is adjusted to the formalisation in this paper.

Definition 6: For a semiotic relation with tolerance relations $T_D \subseteq D \times D$ and $T_I \subseteq I \times I$ the following are defined:

- i_1 and i_2 are *compatible* $\Leftrightarrow \forall_{r \in R, i_1(r) \neq d_\perp, i_2(r) \neq d_\perp} : (i_1(r), i_2(r)) \in T_D$
- i_1 and i_2 are *mergeable* $\Leftrightarrow \forall_{r \in R, i_1(r) \neq d_\perp, i_2(r) \neq d_\perp} : i_1(r) = i_2(r)$
- i_1 and i_2 are *T_I-mergeable* $\Leftrightarrow (i_1, i_2) \in T_I$ and i_1 and i_2 are mergeable
- (i_1, r_1, d_1) and (i_2, r_2, d_2) are *strong synonyms* $\Leftrightarrow r_1 \neq r_2$ and $d_1 = d_2$
- (i_1, r_1, d_1) and (i_2, r_2, d_2) are *synonyms* $\Leftrightarrow r_1 \neq r_2$ and $(d_1, d_2) \in T_D$
- (i_1, r_1, d_1) and (i_2, r_2, d_2) are *equinyms* $\Leftrightarrow r_1 = r_2$ and $d_1 = d_2$
- (i_1, r_1, d_1) and (i_2, r_2, d_2) are *polysemous* $\Leftrightarrow r_1 = r_2$ and $(d_1, d_2) \in T_D$
- (i_1, r_1, d_1) and (i_2, r_2, d_2) are *homographs* $\Leftrightarrow r_1 = r_2$ and $(d_1, d_2) \notin T_D$

It follows that if a representamen lattice is complete for a set of interpretations, representamens that share their object concepts are strong synonyms for all interpretations. In Example 1, if T_D corresponds to $\{$Hello World, How are you$\}$, $\{$yes, no$\}$, $\{1, 2\}$ then firstLoop and secondLoop are compatible. Essentially this means that variables do not radically change their meaning between firstLoop and secondLoop. Mergeable interpretations have the same denotation for each representamen and could be merged into one interpretation. In Example 1 the interpretations in I_A (or in I_B) are not mergeable. Using T_I-mergeability it can be ensured that only interpretations which have something in common (for example temporal adjacency) are merged. There are no examples of homographs in Example 1 but the following table shows some examples for the other notions of Definition 6.

strong synonyms	(firstLoop, input2, "Hello World")	(secondLoop, input1, "Hello World")
synonyms	(firstLoop, input1, "Hello World")	(secondLoop, input2, "How are you")
equinyms	(firstLoop, input1, "Hello World")	(secondLoop, input1, "Hello World")
polysemous	(firstLoop, input2, "Hello World")	(secondLoop, input2, "How are you")

Some programming languages use further types of synonymy-like relations, for example, variables can have the same value and but not the same data type or the same value but not be referring to the same object. An example of homographs in natural languages is presented by the verb 'lead' and the metal 'lead'. In programming languages, homographs are variables which have the same name but are used for totally different purposes. If this happens in separate subroutines of a program, it does not pose a problem. But if it involves global variables it might indicate an error in the code. Thus algorithms for *homograph detection* can be useful for checking the consistency of programs. Compatible interpretations are free of homographs.

Definition 7: A semiotic relation with concept lattices as defined in Definitions 3-5 is called a *semiotic system*. The study of semiotic systems is called a *semiotic-conceptual analysis*.

[4] A tolerance relation is reflexive and symmetric.

5 Mappings between the concept lattices

A next step is to investigate how (and whether) the interpretations as functions from R to D give rise to interesting mappings between the representamen and denotation lattice. For example, if the representamen lattice has an attribute 'starts with uppercase letter' and it is common practice in a programming language to use uppercase letters for names of classes and there is an attribute 'classes' in the denotation lattice, then one would want to investigate whether this information is preserved by the mapping amongst the lattices. The following definition describes a basic relationship:

Definition 8: For a semiotic relation, the power set P(R), subsets $I_1 \subseteq I$ and $R_1 \subseteq R$ we define: $I_1^\vee : P(R)\backslash\{\} \to B(D, M_D, J_D)$ with $I_1^\vee(R_1) := \bigvee_{i\in I_1} \bigvee_{r\in R_1} \gamma(i(r))$.

Because the join relation in a lattice is commutative and associative it does not matter whether one first iterates through interpretations or through representamens (i.e., $\bigvee_{i\in I_1} \bigvee_{r\in R_1}$ or $\bigvee_{r\in R_1} \bigvee_{i\in I_1}$). An analogous function can be defined for infima. One can also consider the inverse $(I_1^\vee)^{-1}$.

Definition 8 allows for different types of applications. One can look at the results for extensions (and thus concepts of the representamen lattice), one-element sets (corresponding to individual elements in R) or elements of a tolerance relation. The same holds for the subsets of I. The question that arises in each application is whether the mapping $I_1^\vee$ has some further properties, such as being order-preserving or whether it forms an 'infomorphism' in Barwise & Seligman's (1997) terminology (together with an inverse mapping). It may be of interest to find the subsets of R for which $(I_1^\vee)^{-1} I_1^\vee(R_1) = R_1$.

In the case of Figure 2, 'input_end' is mapped onto the concepts with attribute 'positive', 'negative' or 'binary' depending on which set of interpretations is used. The other representamens are always mapped onto the same concepts no matter which set of interpretations is used. For the extensions of the representamen lattice this leads to an order-preserving mapping. Thus overall the structures of the representamen and denotation lattice seem very compatible in this example. Other examples could produce mappings which change radically between different interpretations. In a worst case scenario, every representamen is mapped to the top concept of the denotation lattice as soon as more than one interpretation is involved.

In Figure 2 the interpretation lattice is depicted without any connection to the other two lattices. Furthermore even though a construction of $R_1^\vee$ in analogy to $I_1^\vee$ would be possible it would not be interesting for most applications because most elements would be mapped to the top element of the denotation lattice. Thus different strategies are needed for the representamen and interpretation lattices. One possibility of connecting the three lattices is to use a 'faceted display' similar to Priss (2000). The idea for Figure 3 is to use two facets: the denotation lattice which also contains the mapped representamens and the interpretation lattice. If a user 'clicks' on the upper concept in the interpretation lattice, the lattice on the left-hand side of Figure 3 is displayed. If a user clicks on the lower interpretation, the lattice on the right-hand side is displayed. Switching between the two interpretations would show the movement of 'input_end'. This is also reminiscent of the work by Wolff (2004) who uses 'animated' concept lattices which show the movement of 'complex objects' (in contrast to formal objects)

across the nodes of a concept lattice. In our semiotic-conceptual analysis the interpretations are not necessarily linearly-ordered (as Wolff's time units) but ordered according to a concept lattice.

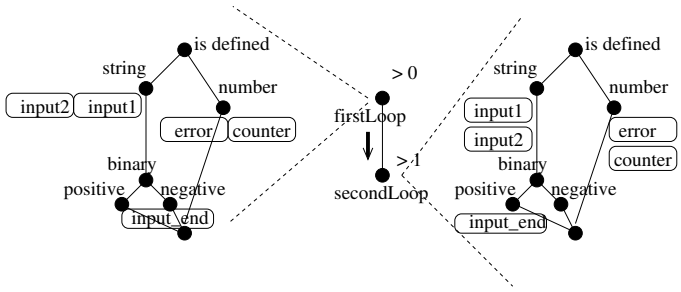


Fig. 3. Switching between interpretations

Instead of variables or strings, representamens can also be more complex structures, such as graphs, UML diagrams, Peirce's existential graphs, relations or other complex mathematical structures which are then analysed using interpretations. Figure 4 shows an example from Priss (1998) which was originally presented in terms of what Priss called 'relational concept analysis[5]'. The words in the figure are entries from the electronic lexical database WordNet[6]. The solid lines in Figure 4 are subconcept relation instances from a concept lattice although the lattice drawing is incomplete in the figure. The dashed lines are part-whole relation instances that are defined among the concepts. Using a semiotic-conceptual analysis, this figure can be generated by using representamens which are instances of a part-whole relation. Two interpretations are involved: one maps the first component of each relation instance into the denotation lattice, the other one maps the second component. Each dashed line corresponds to the mapping of one representamen. For each representamen, $I^{\vee}(r)$ is the whole and $I^{\wedge}(r)$ the part of the relation instance. Priss (1999) calculates bases for semantic relations which in this modelling as a semiotic-conceptual analysis correspond to searching for infima and suprema of such representamens as binary relations.

Figure 4 shows an example of a data error. The supremum of 'hand' and 'foot' should be the concept which is a part of 'limb'. There should be a part-whole relation from 'digit' to that 'extremity' concept. Although it would be possible to write an algorithm that checks for this error systematically, this is probably again an example of where a user can detect an error in the data more easily (because of the lack of symmetry) if the data is graphically presented. We argue that there are so many different ways of how semiotic-conceptual analyses can be used that it is not feasible to write

[5] These days the notion 'relational concept analysis' is used in a different meaning by other authors.

[6] https://wordnet.princeton.edu/

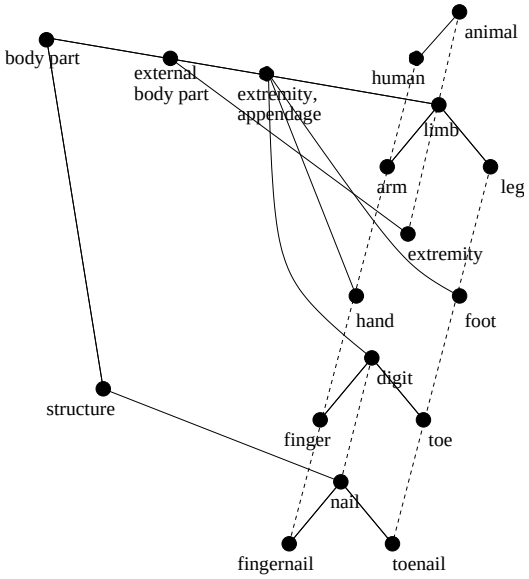


Fig. 4. Representamens showing a part-whole relation

algorithms for any possible situation. In many cases the data can be modelled for an application and then interactively investigated.

In Figure 4, the representamens are instances of a binary relation or pairs of denotations. Thus there are no intrinsic differences between what is a representamen, denotation or interpretation. Denotations are often represented by strings and thus are signs themselves (with respect to another semiotic relation). A computer program as a whole can also be a representamen. Since that is then a single representamen, the relation between the program output (its denotations) and the succession of states (its interpretations) is then a binary relation. Priss (2004) shows an example of a concept lattice for such a relation.

6 Conclusion and outlook

This paper presents a semiotic-conceptual analysis that models the three components of a Peircean semiotic relation as concept lattices which are connected via a semiotic mapping. The paper shows that the formalisation of such a semiotic-conceptual analysis provides a unified framework for a number of our previous FCA applications (Priss, 1998-2004). It also presents another view on Wolff's (2004) animated concept lattices.

But this paper only describes a starting point for this kind of modelling. Instead of considering one semiotic system with sets R, D, I, one could also consider several semiotic systems with sets R_1, D_1, I_1 and so on as subsets of larger sets R, D, I. Then one could investigate what happens if, for example, the signs from one semiotic system become the representamens, interpretations or denotations of another semiotic system.

For example, in the second half of Peirce's sign definition in Section 1 he suggests that for $i_1(r) = d$ there should be an i_2 with $i_2(i_1) = d$. Furthermore one could consider a denotation lattice as a channel between different representamen lattices in the terminology of Barwise & Seligman's (1997) information flow theory as briefly mentioned in Section 5 which also poses some other open questions.

There are connections with existing formalisms (for example model-theoretic semantics) that need further exploration. In some sense a semiotic-conceptual analysis subsumes syntactic relationships (among representamens), semantic relationships (among denotations) and pragmatic relationships (among interpretations) in one formalisation. Other forms of semiotic analyses which use the definitions from Section 2 and 4 but use other structures than concept lattices (as suggested in Section 3) are possible as well. Hopefully future research will address such questions and continue this work.

References

1. Barwise, Jon; Seligman, Jerry (1997). *Information Flow. The Logic of Distributed Systems.* Cambridge University Press.
2. Belohlavek, Radim; Osicka, Petr (2012). *Triadic concept lattices of data with graded attributes.* International Journal of General Systems 41.2, p. 93-108.
3. Ganter, Bernhard; Wille, Rudolf (1999). *Formal Concept Analysis. Mathematical Foundations.* Berlin-Heidelberg-New York: Springer.
4. Gnatyshak, Dmitry; Ignatov, Dmitry; Kuznetsov, Sergei O. (2013). *From Triadic FCA to Triclustering: Experimental Comparison of Some Triclustering Algorithms.* CLA. Vol. 1062.
5. Goguen, Joseph (1999). *An introduction to algebraic semiotics, with application to user interface design.* Computation for metaphors, analogy, and agents. Springer Berlin Heidelberg, p. 242-291.
6. Priss, Uta (1998). *The Formalization of WordNet by Methods of Relational Concept Analysis.* In: Fellbaum, Christiane (ed.), WordNet: An Electronic Lexical Database and Some of its Applications, MIT press, p. 179-196.
7. Priss, Uta (1999). *Efficient Implementation of Semantic Relations in Lexical Databases.* Computational Intelligence, Vol. 15, 1, p. 79-87.
8. Priss, Uta (2000). *Lattice-based Information Retrieval.* Knowledge Organization, Vol. 27, 3, p. 132-142.
9. Priss, Uta (2004). *Signs and Formal Concepts.* In: Eklund (ed.), Concept Lattices: Second International Conference on Formal Concept Analysis, Springer Verlag, LNCS 2961, 2004, p. 28-38.
10. Maddux, Roger D. (1991). *The origin of relation algebras in the development and axiomatization of the calculus of relations.* Studia Logica 50, 3-4, p. 421-455.
11. Marty, Robert (1992). *Foliated semantic networks: concepts, facts, qualities.* Computers & mathematics with applications 23.6, p. 679-696.
12. Wolff, Karl Erich (2004). *Towards a conceptual theory of indistinguishable objects.* Concept Lattices. Springer Berlin Heidelberg, p. 180-188.
13. Zalamea, Fernando (2010). *Towards a Complex Variable Interpretation of Peirces Existential Graphs.* In: Bergman, M., Paavola, S., Pietarinen, A.-V., & Rydenfelt, H. (Eds.). Ideas in Action: Proceedings of the Applying Peirce Conference, p. 277-287.

Using the Chu construction for generalizing formal concept analysis

L. Antoni[1], I.P. Cabrera[2], S. Krajči[1], O. Krídlo[1], M. Ojeda-Aciego[2]

[1] University of Pavol Jozef Šafárik, Košice, Slovakia *
[2] Universidad de Málaga. Departamento Matemática Aplicada. Spain **

Abstract. The goal of this paper is to show a connection between FCA generalisations and the Chu construction on the category ChuCors, the category of formal contexts and Chu correspondences. All needed categorical properties like categorical product, tensor product and its bifunctor properties are presented and proved. Finally, the second order generalisation of FCA is represented by a category built up in terms of the Chu construction.

Keywords: formal concept analysis, category theory, Chu construction

1 Introduction

The importance of category theory as a foundational tool was discovered soon after its very introduction by Eilenberg and MacLane about seventy years ago. On the other hand, Formal Concept Analysis (FCA) has largely shown both its practical applications and its capability to be generalized to more abstract frameworks, and this is why it has become a very active research topic in the recent years; for instance, a framework for FCA has been recently introduced in [19] in which the sets of objects and attributes are no longer unstructured but have a hypergraph structure by means of certain ideas from mathematical morphology. On the other hand, for an application of the FCA formalism to other areas, in [11] the authors introduce a representation of algebraic domains in terms of FCA.

The Chu construction [8] is a theoretical method that, from a symmetric monoidal closed (autonomous) category and a dualizing object, generates a *-autonomous category. This construction, or the closely related notion of Chu space, has been applied to represent quantum physical systems and their symmetries [1, 2].

This paper continues with the study of the categorical foundations of formal concept analysis. Some authors have noticed the property of being a cartesian closed category of certain concept structures that can be approximated [10, 20];

* Partially supported by the Scientific Grant Agency of the Ministry of Education of Slovak Republic under contract VEGA 1/0073/15.
** Partially supported by the Spanish Science Ministry projects TIN12-39353-C04-01 and TIN11-28084.

others have provided a categorical construction of certain extensions of FCA [12]; morphisms have received a categorical treatment in [17] as a means for the modelling of communication.

There already exist some approaches [9] which consider the Chu construction in terms of FCA. In the current paper, we continue the previous study by the authors on the categorical foundation of FCA [13,15,16]. Specifically, the goal of this paper is to highlight the importance of the Chu construction in the research area of categorical description of the theory of FCA and its generalisations. The Chu construction plays here the role of some recipe for constructing a suitable category that covers the second order generalisation of FCA.

The structure of this paper is the following: in Section 2 we recall the preliminary notions required both from category theory and formal concept analysis. Then, the various categorical properties of the input category which are required (like the existence of categorical and tensor product) are developed in detail in Sections 3 and 4. An application of the Chu construction is presented in Section 5 where it is also showed how to construct formal contexts of second order from the category of classical formal contexts and Chu correspondences (ChuCors).

2 Preliminaries

In order to make the manuscript self-contained, the fundamental notions and its required properties are recalled in this section.

Definition 1. *A* formal context *is any triple* $\mathcal{C} = \langle \mathcal{B}, \mathcal{A}, \mathcal{R} \rangle$ *where* B *and* A *are finite sets and* $R \subseteq B \times A$ *is a binary relation. It is customary to say that* B *is a set of* objects*,* A *is a set of* attributes *and* R *represents a relation between objects and attributes.*

On a given formal context (B, A, R)*, the derivation (or concept-forming) operators are a pair of mappings* $\uparrow\colon 2^B \to 2^A$ *and* $\downarrow\colon 2^A \to 2^B$ *such that if* $X \subseteq B$*, then* $\uparrow X$ *is the set of all attributes which are related to every object in* X *and, similarly, if* $Y \subseteq A$*, then* $\downarrow Y$ *is the set of all objects which are related to every attribute in* Y*.*

In order to simplify the description of subsequent computations, it is convenient to describe the concept forming operators in terms of characteristic functions, namely, considering the subsets as functions on the set of Boolean values. Specifically, given $X \subseteq B$ and $Y \subseteq A$, we can consider mappings $\uparrow X\colon A \to \{0,1\}$ and $\downarrow Y\colon B \to \{0,1\}$

1. $\uparrow X(a) = \bigwedge_{b \in B} \big((b \in X) \Rightarrow ((b,a) \in R)\big)$ for any $a \in A$
2. $\downarrow Y(b) = \bigwedge_{a \in A} \big((a \in Y) \Rightarrow ((b,a) \in R)\big)$ for any $b \in B$

where the infimum is considered in the set of Boolean values and $\Rightarrow$ is the truth-function of the implication of classical logic.

Definition 2. *A* formal concept *is a pair of sets* $\langle X, Y\rangle \in 2^B \times 2^A$ *which is a fixpoint of the pair of concept-forming operators, namely,* $\uparrow X = Y$ *and* $\downarrow Y = X$. *The object part* X *is called the* extent *and the attribute part* Y *is called the* intent.

There are two main constructions relating two formal contexts: the bonds and the Chu correspondences. Their formal definitions are recalled below:

Definition 3. *Consider* $\mathcal{C}_1 = \langle B_1, A_1, R_1\rangle$ *and* $\mathcal{C}_2 = \langle B_2, A_2, R_2\rangle$ *two formal contexts. A* bond *between* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *is any relation* $\beta \in 2^{B_1 \times A_2}$ *such that its columns are extents of* $\mathcal{C}_1$ *and its rows are intents of* $\mathcal{C}_2$. *All bonds between such contexts will be denoted by* $\mathrm{Bonds}(\mathcal{C}_1, \mathcal{C}_2)$.

The Chu correspondence between contexts can be seen as an alternative inter-contextual structure which, instead, links intents of $\mathcal{C}_1$ and extents of $\mathcal{C}_2$. Namely,

Definition 4. *Consider* $\mathcal{C}_1 = \langle B_1, A_1, R_1\rangle$ *and* $\mathcal{C}_2 = \langle B_2, A_2, R_2\rangle$ *two formal contexts. A* Chu correspondence *between* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *is any pair of multimappings* $\varphi = \langle \varphi_L, \varphi_R\rangle$ *such that*

- $\varphi_L \colon B_1 \to \mathrm{Ext}(\mathcal{C}_2)$
- $\varphi_R \colon A_2 \to \mathrm{Int}(\mathcal{C}_1)$
- $\uparrow_2(\varphi_L(b_1))(a_2) = \downarrow_1(\varphi_R(a_2))(b_1)$ *for any* $(b_1, a_2) \in B_1 \times A_2$

All Chu correspondences between such contexts will be denoted by $\mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2)$.

The notions of bond and Chu correspondence are interchangeable; specifically, we will use the bond β_φ associated to a Chu correspondence φ from $\mathcal{C}_1$ to $\mathcal{C}_2$ defined for $b_1 \in B_1, a_2 \in A_2$ as follows:

$$\beta_\varphi(b_1, a_2) = \uparrow_2 (\varphi_L(b_1))(a_2) = \downarrow_1 (\varphi_R(a_2))(b_1)$$

The set of all bonds (resp. Chu correspondences) between any two formal contexts endowed with set inclusion as ordering have a complete lattice structure. Moreover, both complete lattices are dually isomorphic.

In order to formally define the composition of two Chu correspondences, we need to introduce the extension principle below:

Definition 5. *Given a mapping* $\varphi \colon X \to 2^Y$ *we define its extended mapping* $\varphi_+ \colon 2^X \to 2^Y$ *defined by* $\varphi_+(M) = \bigcup_{x \in M} \varphi(x)$, *for all* $M \in 2^X$.

The set of formal contexts together with Chu correspondences as morphisms forms a category denoted by ChuCors. Specifically:

- *objects* formal contexts
- *arrows* Chu correspondences
- *identity arrow* $\iota \colon \mathcal{C} \to \mathcal{C}$ of context $\mathcal{C} = \langle B, A, R\rangle$
  - $\iota_L(o) = \downarrow\uparrow(\{b\})$, for all $b \in B$
  - $\iota_R(a) = \uparrow\downarrow(\{a\})$, for all $a \in A$

- *composition* $\varphi_2 \circ \varphi_1 \colon \mathcal{C}_1 \to \mathcal{C}_3$ *of arrows* $\varphi_1 \colon \mathcal{C}_1 \to \mathcal{C}_2$, $\varphi_2 \colon \mathcal{C}_2 \to \mathcal{C}_3$ (where $\mathcal{C}_i = \langle B_i, A_i, R_i \rangle$, $i \in \{1,2,3\}$)
  - $(\varphi_2 \circ \varphi_1)_L : B_1 \to 2^{B_3}$ and $(\varphi_2 \circ \varphi_1)_R : A_3 \to 2^{A_1}$
  - $(\varphi_2 \circ \varphi_1)_L(b_1) = \downarrow_3 \uparrow_3 (\varphi_{2L+}(\varphi_{1L}(b_1)))$
  - $(\varphi_2 \circ \varphi_1)_R(a_3) = \uparrow_1 \downarrow_1 (\varphi_{1R+}(\varphi_{2R}(a_3)))$

The category ChuCors is *-autonomous and equivalent to the category of complete lattices and isotone Galois connection, more results on this category and its L-fuzzy extensions can be found in [13,15,16,18].

3 Categorical product on ChuCors

In this section, the category ChuCors is proved to contain all finite categorical products, that is, it is a Cartesian category. To begin with, it is convenient to recall the notion of categorical product.

Definition 6. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two objects in a category. By a* product *of $\mathcal{C}_1$ and $\mathcal{C}_2$ we mean an object $\mathcal{P}$ with arrows $\pi_i \colon \mathcal{P} \to \mathcal{C}_i$ for $i \in \{1,2\}$ satisfying the following condition: For any object $\mathcal{D}$ and arrows $\delta_i \colon \mathcal{D} \to \mathcal{C}_i$ for $i \in \{1,2\}$, there exists a unique arrow $\gamma \colon \mathcal{D} \to \mathcal{P}$ such that $\gamma \circ \pi_i = \delta_i$ for all $i \in \{1,2\}$.*

The construction will use the notion of disjoint union of two sets $S_1 \uplus S_2$ which can be formally described as $(\{1\} \times S_1) \cup (\{2\} \times S_2)$ and, therefore, their elements will be denoted as ordered pairs (i, s) where $i \in \{1,2\}$ and $s \in S_i$. Now, we can proceed with the construction:

Definition 7. *Consider $\mathcal{C}_1 = \langle B_1, A_1, R_1 \rangle$ and $\mathcal{C}_2 = \langle B_2, A_2, R_2 \rangle$ two formal contexts. The* product *of such contexts is a new formal context*

$$\mathcal{C}_1 \times \mathcal{C}_2 = \langle B_1 \uplus B_2, A_1 \uplus A_2, R_{1\times 2} \rangle$$

where the relation $R_{1\times 2}$ is given by

$$((i,b),(j,a)) \in R_{1\times 2} \text{ if and only if } \big((i=j) \Rightarrow (b,a) \in R_i\big)$$

for any $(b,a) \in B_i \times A_j$ and $(i,j) \in \{1,2\} \times \{1,2\}$.

Lemma 1. *The above defined contextual product fulfills the property of the categorical product on the category* ChuCors*.*

Proof. We define the projection arrows $\langle \pi_{iL}, \pi_{iR} \rangle \in \mathrm{Chu}(\mathcal{C}_1 \times \mathcal{C}_2, \mathcal{C}_i)$ for $i \in \{1,2\}$ as follows

- $\pi_{iL} \colon B_1 \uplus B_2 \to \mathrm{Ext}(\mathcal{C}_i) \subseteq 2^{B_i}$
- $\pi_{iR} \colon A_i \to \mathrm{Int}(\mathcal{C}_1 \times \mathcal{C}_2) \subseteq 2^{A_1 \cup A_2}$
- such that for any $(k,x) \in B_1 \uplus B_2$ and $a_i \in A_i$ the following equality holds

$$\uparrow_i (\pi_{iL}(k,x))(a_i) = \downarrow_{1\times 2} (\pi_{iR}(a_i))(k,x)$$

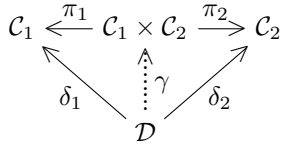
The definition of the projections is given below

$$\pi_{iL}(k,x)(b_i) = \begin{cases} \downarrow_i\uparrow_i(\chi_x)(b_i) \text{ for } k = i \\ \downarrow_i\uparrow_i(\overline{0})(b_i) \text{ for } k \neq i \end{cases} \text{ for any } (k,x) \in B_1 \uplus B_2 \text{ and } b_i \in B_i$$

$$\pi_{iR}(a_i)(k,y) = \begin{cases} \uparrow_i\downarrow_i(\chi_{a_i})(y) \text{ for } k = i \\ \uparrow_k\downarrow_k(\overline{0})(y) \text{ for } k \neq i \end{cases} \text{ for any } (k,y) \in A_1 \uplus A_2 \text{ and } a_i \in A_i.$$

The proof that the definitions above actually provide a Chu correspondence is just a long, although straightforward, computation and it is omitted.

Now, one has to show that to any formal context $\mathcal{D} = \langle E, F, G\rangle$, where $G \subseteq E \times F$ and any pair of arrows (δ_1, δ_2) with $\delta_i \colon \mathcal{D} \to \mathcal{C}_i$ for all $i \in \{1,2\}$, there exists a unique morphism $\gamma \colon \mathcal{D} \to \mathcal{C}_1 \times \mathcal{C}_2$ such that the following diagram commutes:



We give just the definition of γ as a pair of mappings $\gamma_L \colon E \to 2^{B_1 \uplus B_2}$ and $\gamma_R \colon A_1 \uplus A_2 \to 2^F$

- $\gamma_L(e)(k,x) = \delta_{kL}(e)(x)$ for any $e \in E$ and $(k,x) \in B_1 \uplus B_2$.
- $\gamma_R(k,y)(f) = \delta_{kR}(y)(f)$ for any $f \in F$ and $(k,y) \in A_1 \uplus A_2$.

Checking the condition of categorical product is again straightforward but long and tedious and, hence, it is omitted. □

We have just proved that binary products exist, but a cartesian category requires the existence of *all finite products*. If we recall the well-known categorical theorem which states that if a category has a terminal object and binary product, then it has all finite products, we have just to prove the existence of a terminal object (namely, the nullary product) in order to prove ChuCors to be cartesian.

Any formal context of the form $\langle B, A, B \times A\rangle$ where the incidence relation is the full cartesian product of the sets of objects and attributes is (isomorphic to) the terminal object of ChuCors. Such formal context has just one formal concept $\langle B, A\rangle$; hence, from any other formal context there is just one Chu correspondence to $\langle B, A, B \times A\rangle$.

4 Tensor product and its bifunctor property

Apart from the categorical product, another product-like construction can be given in the category ChuCors, for which the notion of transposed context $\mathcal{C}^*$ is needed.

Given a formal context $\mathcal{C} = \langle B, A, R\rangle$, its transposed context is $\mathcal{C}^* = \langle A, B, R^t\rangle$, where $R^t(a,b)$ holds iff $R(b,a)$ holds. Now, if $\varphi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2)$, one can consider $\varphi^* \in \mathrm{Chu}(C_2^*, C_1^*)$ defined by $\varphi_L^* = \varphi_R$ and $\varphi_R^* = \varphi_L$.

Definition 8. *The tensor product of formal contexts* $\mathcal{C}_i = \langle B_i, A_i, R_i \rangle$ *for* $i \in \{1,2\}$ *is defined as the formal context* $\mathcal{C}_1 \boxtimes \mathcal{C}_2 = \langle B_1 \times B_2, \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*), R_\boxtimes \rangle$ *where*

$$R_\boxtimes((b_1,b_2),\varphi) = \downarrow_2(\varphi_L(b_1))(b_2).$$

Mori studied in [18] the properties of the tensor product above, and proved that ChuCors with $\boxtimes$ is a symmetric and monoidal category. Those results were later extended to the L-fuzzy case in [13]. In both papers, the structure of the formal concepts of a product context was established as an ordered pair formed by a bond and a set of Chu correspondences.

Lemma 2. *Let* $\mathcal{C}_i = \langle B_i, A_i, R_i \rangle$ *for* $i \in \{1,2\}$ *be two formal contexts, and let* $\langle \beta, X \rangle \in \mathrm{Bonds}(\mathcal{C}_1, \mathcal{C}_2^*) \times 2^{\mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*)}$ *be an arbitrary formal concept of* $\mathcal{C}_1 \boxtimes \mathcal{C}_2$. *Then* $\beta = \bigwedge_{\psi \in X} \beta_\psi$ *and* $X = \{\psi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*) \mid \beta \leq \beta_\psi\}$.

Proof. Let X be an arbitrary subset of $\mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*)$. Then, for all $(b_1, b_2) \in B_1 \times B_2$, we have

$$\begin{aligned}\downarrow_{\mathcal{C}_1 \boxtimes \mathcal{C}_2}(X)(b_1,b_2) &= \bigwedge_{\psi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*)} \big((\psi \in X) \Rightarrow \downarrow_2(\psi_L(b_1))(b_2)\big) \\ &= \bigwedge_{\psi \in X} \downarrow_2(\psi_L(b_1))(b_2) = \bigwedge_{\psi \in X} \beta_\psi(b_1,b_2)\end{aligned}$$

Let β be an arbitrary subset of $B_1 \times B_2$. Then, for all $\psi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*)$

$$\begin{aligned}\uparrow_{\mathcal{C}_1 \boxtimes \mathcal{C}_2}(\beta)(\psi) &= \bigwedge_{(b_1,b_2) \in B_1 \times B_2} \big(\beta(b_1,b_2) \Rightarrow \downarrow_2(\psi_L(b_1))(b_2)\big) \\ &= \bigwedge_{(b_1,b_2) \in B_1 \times B_2} \big(\beta(b_1,b_2) \Rightarrow \beta_\psi(b_1,b_2)\big)\end{aligned}$$

Hence $\uparrow_{\mathcal{C}_1 \boxtimes \mathcal{C}_2}(\beta) = \{\psi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*) \mid \beta \leq \beta_\psi\}$ □

We now introduce the notion of product of one context with a Chu correspondence.

Definition 9. *Let* $\mathcal{C}_i = \langle B_i, A_i, R_i \rangle$ *for* $i \in \{0,1,2\}$ *be formal contexts, and consider* $\varphi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2)$. *Then, the pair of mappings*

$$(\mathcal{C}_0 \boxtimes \varphi)_L \colon B_0 \times B_1 \to 2^{B_0 \times B_2} \qquad (\mathcal{C}_0 \boxtimes \varphi)_R \colon \mathrm{Chu}(\mathcal{C}_0, \mathcal{C}_2) \to 2^{\mathrm{Chu}(\mathcal{C}_0, \mathcal{C}_1)}$$

is defined as follows:

- $(\mathcal{C}_0 \boxtimes \varphi)_L(b,b_1)(o,b_2) = \downarrow_{\mathcal{C}_0 \boxtimes \mathcal{C}_2} \uparrow_{\mathcal{C}_0 \boxtimes \mathcal{C}_2} (\gamma_\varphi^{b,b_1})(o,b_2)$ *where* $\gamma_\varphi^{b,b_1}(o,b_2) = \big((b = o) \wedge \varphi_L(b_1)(b_2)\big)$ *for any* $b, o \in B_0$, $b_i \in B_i$ *with* $i \in \{1,2\}$
- $(\mathcal{C}_0 \boxtimes \varphi)_R(\psi_2)(\psi_1) = \big(\psi_1 \leq (\psi_2 \circ \varphi^*)\big)$ *for any* $\psi_i \in \mathrm{Chu}(\mathcal{C}_0, \mathcal{C}_i)$

As one could expect, the result is a Chu correspondence between the products of the contexts. Specifically:

Lemma 3. *Let $\mathcal{C}_i = \langle B_i, A_i, R_i\rangle$ be formal contexts for $i \in \{0,1,2\}$, and consider $\varphi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2)$. Then $\mathcal{C}_0 \boxtimes \varphi \in \mathrm{Chu}(\mathcal{C}_0 \boxtimes \mathcal{C}_1, \mathcal{C}_0 \boxtimes \mathcal{C}_2)$.*

Proof. $(\mathcal{C}_0 \boxtimes \varphi)_L(b, b_1) \in \mathrm{Ext}(\mathcal{C}_0 \boxtimes \mathcal{C}_2)$ for any $(b, b_1) \in B_0 \times B_1$ follows directly from its definition. $(\mathcal{C}_0 \boxtimes \varphi)_R(\psi) \in \mathrm{Int}(\mathcal{C}_0 \boxtimes \mathcal{C}_1)$ for any $\psi \in \mathrm{Chu}(\mathcal{C}_0, \mathcal{C}_1)$ follows from Lemma 2.

Consider an arbitrary $b \in B_0$, $b_1 \in B_1$ and $\psi_2 \in \mathrm{Chu}(\mathcal{C}_0, \mathcal{C}_2^*)$

$$
\begin{aligned}
&\uparrow_{\mathcal{C}_0\boxtimes\mathcal{C}_2}\big((\mathcal{C}_0 \boxtimes \varphi)_L(b, b_1)\big)(\psi_2)\\
&\quad= \uparrow_{\mathcal{C}_0\boxtimes\mathcal{C}_2}\downarrow_{\mathcal{C}_0\boxtimes\mathcal{C}_2}\uparrow_{\mathcal{C}_0\boxtimes\mathcal{C}_2}(\gamma_\varphi^{b,b_1})(\psi_2)\\
&\quad= \uparrow_{\mathcal{C}_0\boxtimes\mathcal{C}_2}(\gamma_\varphi^{b,b_1})(\psi_2)\\
&\quad= \bigwedge_{(o,b_2)\in B_0\times B_2} \big(\gamma_\varphi^{b,b_1}(o, b_2) \Rightarrow \downarrow(\psi_{2R}(b_2))(o)\big)\\
&\quad= \bigwedge_{(o,b_2)\in B_0\times B_2} \Big(\big((o = b) \wedge \varphi_L(b_1)(b_2)\big) \Rightarrow \downarrow(\psi_{2R}(b_2))(o)\Big)\\
&\quad= \bigwedge_{o\in B_0}\bigwedge_{b_2\in B_2} \Big((o = b) \Rightarrow \big(\varphi_L(b_1)(b_2) \Rightarrow \downarrow(\psi_{2R}(b_2))(o)\big)\Big)\\
&\quad= \bigwedge_{o\in B_0} \Big((o = b) \Rightarrow \bigwedge_{b_2\in B_2} (\varphi_L(b_1)(b_2) \Rightarrow \downarrow(\psi_{2R}(b_2))(o))\Big)\\
&\quad= \bigwedge_{b_2\in B_2} \big(\varphi_L(b_1)(b_2) \Rightarrow \downarrow(\psi_{2R}(b_2))(b)\big)\\
&\quad= \bigwedge_{b_2\in B_2} \Big(\varphi_L(b_1)(b_2) \Rightarrow \bigwedge_{a\in A} (\psi_{2R}(b_2)(a) \Rightarrow R(b, a))\Big)\\
&\quad= \bigwedge_{a\in A} \Big(\bigvee_{b_2\in B_2} (\varphi_L(b_1)(b_2) \wedge \psi_{2R}(b_2)(a)) \Rightarrow R(b, a)\Big)\\
&\quad= \bigwedge_{a\in A} \Big(\big(\psi_{2R+}(\varphi_L(b_1))(a)\big) \Rightarrow R(b, a)\Big)\\
&\quad= \downarrow(\psi_{2R+}(\varphi_L(b_1))(b) = \downarrow\uparrow\downarrow(\psi_{2R+}(\varphi_L(b_1))(b) = \downarrow((\varphi \circ \psi_2)_R(b_1))(b)
\end{aligned}
$$

Note the use above of the extended mapping as given in Definition 5 in relation to the composition of Chu correspondences.

On the other hand, we have

$$
\begin{aligned}
&\downarrow_{\mathcal{C}_0\boxtimes\mathcal{C}_1}((\mathcal{C}_0 \boxtimes \varphi)_R(\psi_2))(b, b_1)\\
&\quad= \bigwedge_{\psi_1\in\mathrm{Chu}(\mathcal{C}_0,\mathcal{C}_1)} ((\mathcal{C}_0 \boxtimes \varphi)_R(\psi_2)(\psi_1) \Rightarrow \downarrow(\psi_{1R}(b_1))(b))\\
&\quad= \bigwedge_{\psi_1\in\mathrm{Chu}(\mathcal{C}_0,\mathcal{C}_1)} ((\psi_1 \geq \varphi \circ \psi_2) \Rightarrow \downarrow(\psi_{1R}(b_1))(b))
\end{aligned}
$$

$$= \bigwedge_{\substack{\psi_1 \in \mathrm{Chu}(\mathcal{C}_0,\mathcal{C}_1) \\ \psi_1 \geq \varphi \circ \psi_2}} \downarrow(\psi_{1R}(b_1))(b) = \downarrow((\varphi \circ \psi_2)_R(b_1))(b)$$

Hence $\uparrow_{\mathcal{C}_0 \boxtimes \mathcal{C}_2} ((\mathcal{C}_0 \boxtimes \varphi)_L(b, b_1))(\psi_2) = \downarrow_{\mathcal{C}_0 \boxtimes \mathcal{C}_1} ((\mathcal{C}_0 \boxtimes \varphi)_R(\psi_2))(b, b_1)$. So if $\varphi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2)$ then $\mathcal{C}_0 \boxtimes \varphi \in \mathrm{Chu}(\mathcal{C}_0 \boxtimes \mathcal{C}_1, \mathcal{C}_0 \boxtimes \mathcal{C}_2)$. □

Given a fixed formal context $\mathcal{C}$, the tensor product $\mathcal{C} \boxtimes (-)$ forms a mapping between objects of ChuCors assigning to any formal context $\mathcal{D}$ the formal context $\mathcal{C} \boxtimes \mathcal{D}$. Moreover to any arrow $\varphi \in \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2)$ it assigns an arrow $\mathcal{C} \boxtimes \varphi \in \mathrm{Chu}(\mathcal{C} \boxtimes \mathcal{C}_1, \mathcal{C} \boxtimes \mathcal{C}_2)$. We will show that this mapping preserves the unit arrows and the composition of Chu correspondences. Hence the mapping forms an endofunctor on ChuCors, that is, a covariant functor from the category ChuCors to itself.

To begin with, let us recall the definition of functor between two categories:

Definition 10 (See [6]). *A covariant functor* $F\colon \mathrm{C} \to \mathrm{D}$ *between categories* C *and* D *is a mapping of objects to objects and arrows to arrows, in such a way that:*

- *For any morphism* $f\colon A \to B$*, one has* $F(f)\colon F(A) \to F(B)$
- $F(g \circ f) = F(g) \circ F(f)$
- $F(1_A) = 1_{F(A)}$.

Lemma 4. *Let* $\mathcal{C} = \langle B, A, R \rangle$ *be a formal context.* $\mathcal{C} \boxtimes (-)$ *is an endofunctor on* ChuCors.

Proof. Consider the unit morphism $\iota_{\mathcal{C}_1}$ of a formal context $\mathcal{C}_1 = \langle B_1, A_1, R_1 \rangle$, and let us show that $(\mathcal{C} \boxtimes \iota_{\mathcal{C}_1}) = \iota_{\mathcal{C} \boxtimes \mathcal{C}_1}$. In other words, $\mathcal{C} \boxtimes (-)$ respects unit arrows in ChuCors.

$$\begin{aligned}
&\uparrow_{\mathcal{C} \boxtimes \mathcal{C}_1} \big((\mathcal{C} \boxtimes \iota_{\mathcal{C}_1})(b, b_1)\big)(\psi) \\
&\quad = \bigwedge_{(o,o_1) \in B \times B_1} \Big(\big((o = b) \wedge \iota_{\mathcal{C}_1 L}(b_1)(o_1)\big) \Rightarrow \downarrow_1(\psi_L(o))(o_1)\Big) \\
&\quad = \bigwedge_{o_1 \in B_1} \big(\downarrow_1 \uparrow_1 (\chi_{b_1})(o_1) \Rightarrow \downarrow_1(\psi_L(b))(o_1)\big) \\
&\quad = \bigwedge_{o_1 \in B_1} \Big(\downarrow_1 \uparrow_1 (\chi_{b_1})(o_1) \Rightarrow \bigwedge_{a_1 \in A_1} \big(\psi_L(b)(a_1) \Rightarrow R(o_1, a_1)\big)\Big) \\
&\quad = \bigwedge_{o_1 \in B_1} \bigwedge_{a_1 \in A_1} \Big(\downarrow_1 \uparrow_1 (\chi_{b_1})(o_1) \Rightarrow \big(\psi_L(b)(a_1) \Rightarrow R(o_1, a_1)\big)\Big) \\
&\quad = \bigwedge_{o_1 \in B_1} \bigwedge_{a_1 \in A_1} \Big(\psi_L(b)(a_1) \Rightarrow \big(\downarrow_1 \uparrow_1 (\chi_{b_1})(o_1) \Rightarrow R(o_1, a_1)\big)\Big) \\
&\quad = \bigwedge_{a_1 \in A_1} \Big(\psi_L(b)(a_1) \Rightarrow \bigwedge_{o_1 \in B_1} \big(\downarrow_1 \uparrow_1 (\chi_{b_1})(o_1) \Rightarrow R(o_1, a_1)\big)\Big) \\
&\quad = \bigwedge_{a_1 \in A_1} \big(\psi_L(b)(a_1) \Rightarrow \uparrow_1 \downarrow_1 \uparrow_1 (\chi_{b_1})(a_1)\big)
\end{aligned}$$

$$= \bigwedge_{a_1 \in A_1} \big(\psi_L(b)(a_1) \Rightarrow R_1(b_1, a_1)\big) = \downarrow_1 (\psi_L(b))(b_1)$$

and, on the other hand, we have

$$\begin{aligned}
&\uparrow_{\mathcal{C}\boxtimes\mathcal{C}_1}(\iota_{\mathcal{C}\boxtimes\mathcal{C}_1}(b, b_1))(\psi) \\
&\quad = \uparrow_{\mathcal{C}\boxtimes\mathcal{C}_1}(\chi_{(b,b_1)})(\psi) \\
&\quad = \bigwedge_{(o,o_1)\in B\times B_1} \big(\chi_{(b,b_1)}(o, o_1) \Rightarrow \downarrow_1 (\psi_L(o))(o_1)\big) \\
&\quad = \downarrow_1 (\psi_L(b))(b_1)
\end{aligned}$$

As a result, we have obtained $\uparrow_{\mathcal{C}\boxtimes\mathcal{C}_1}((\mathcal{C}\boxtimes\iota_{\mathcal{C}_1})(b, b_1))(\psi) = \uparrow_{\mathcal{C}\boxtimes\mathcal{C}_1}(\iota_{\mathcal{C}\boxtimes\mathcal{C}_1}(b, b_1))(\psi)$ for any $(b, b_1) \in B \times B_1$ and any $\psi \in \mathrm{Chu}(\mathcal{C}, \mathcal{C}_1)$; hence, $\iota_{\mathcal{C}\boxtimes\mathcal{C}_1} = (\mathcal{C} \boxtimes \iota_{\mathcal{C}_1})$.

We will show now that $\mathcal{C} \boxtimes (-)$ preserves the composition of arrows. Specifically, this means that for any two arrows $\varphi_i \in \mathrm{Chu}(\mathcal{C}_i, \mathcal{C}_{i+1})$ for $i \in \{1, 2\}$ it holds that $\mathcal{C} \boxtimes (\varphi_1 \circ \varphi_2) = (\mathcal{C} \boxtimes \varphi_1) \circ (\mathcal{C} \boxtimes \varphi_2)$.

$$\begin{aligned}
&\uparrow_{\mathcal{C}\boxtimes\mathcal{C}_3}\big(\big(\mathcal{C} \boxtimes (\varphi_1 \circ \varphi_2)\big)_L(b, b_1)\big)(\psi_3) \\
&\quad = \bigwedge_{(o,b_3)\in B\times B_3} \Big(\big((o = b) \wedge (\varphi_1 \circ \varphi_2)_L(b_1)(b_3)\big) \Rightarrow \downarrow(\psi_{3R}(b_3))(o)\Big) \\
&\quad = \bigwedge_{b_3\in B_3} \big((\varphi_1 \circ \varphi_2)_L(b_1)(b_3) \Rightarrow \downarrow(\psi_{3R}(b_3))(b)\big) \\
&\qquad \text{(by similar operations to those in the first part of the proof)} \\
&\quad = \downarrow\big(\big((\varphi_1 \circ \varphi_2) \circ \psi_3\big)_L(b_1)\big)(b)
\end{aligned}$$

On the other hand, and writing F for $\mathcal{C} \boxtimes -$ in order to simplify the resulting expressions, we have

$$\begin{aligned}
&\uparrow_{F\mathcal{C}_3}((F\varphi_1 \circ F\varphi_2)_L(b, b_1))(\psi_3) \\
&= \uparrow_{F\mathcal{C}_3}\downarrow_{F\mathcal{C}_3}\uparrow_{F\mathcal{C}_3}\big((F\varphi_2)_{L+}\big((F\varphi_1)_L(b, b_1)\big)\big)(\psi_3) \\
&= \bigwedge_{(o,b_3)\in B\times B_3} \Bigg(\\
&\qquad \bigvee_{(j,b_2)\in B\times B_2} \Big((F\varphi_1)_L(b, b_1)(j, b_2) \wedge (F\varphi_2)_L(j, b_2)(o, b_3)\Big) \Rightarrow \downarrow(\psi_{3R}(b_3))(o)\Bigg) \\
&= \bigwedge_{b_3\in B_3} \bigwedge_{b_2\in B_2} \Big(\big(\varphi_{1L}(b_1)(b_2) \wedge \varphi_{2L}(b_2)(b_3)\big) \Rightarrow \downarrow(\psi_{3R}(b_3))(b)\Big) \\
&= \bigwedge_{b_3\in B_3} \Big(\bigvee_{b_2\in B_2} \big(\varphi_{1L}(b_1)(b_2) \wedge \varphi_{2L}(b_2)(b_3)\big) \Rightarrow \downarrow(\psi_{3R}(b_3))(b)\Big) \\
&= \bigwedge_{b_3\in B_3} \Big(\varphi_{2L+}(\varphi_{1L}(b_1))(b_3) \Rightarrow \downarrow(\psi_{3R}(b_3))(b)\Big)
\end{aligned}$$

$$= \bigwedge_{b_3 \in B_3} \Big((\varphi_1 \circ \varphi_2)_L(b_1)(b_3) \Rightarrow \downarrow(\psi_{3R}(b_3))(b) \Big)$$

From the previous equalities we see that $\mathcal{C} \boxtimes (\varphi_1 \circ \varphi_2) = (\mathcal{C} \boxtimes \varphi_1) \circ (\mathcal{C} \boxtimes \varphi_2)$. Hence, composition is preserved.

As a result, the mapping $\mathcal{C} \boxtimes (-)$ forms a functor from ChuCors to itself. □

All the previous computations can be applied to the first argument without any problems, hence we can directly state the following proposition.

Proposition 1. *The tensor product forms a bifunctor* $- \boxtimes -$ *from* ChuCors $\times$ ChuCors *to* ChuCors.

5 The Chu construction on ChuCors and second order formal concept analysis

A second order formal context [14] focuses on the external formal contexts and it serves a bridge between the L-fuzzy [3, 7] and heterogeneous [4] frameworks.

Definition 11. *Consider two non-empty index sets* I *and* J *and an* L*-fuzzy formal context* $\langle \bigcup_{i \in I} B_i, \bigcup_{j \in J} A_j, r \rangle$*, whereby*

- $B_{i_1} \cap B_{i_2} = \emptyset$ *for any* $i_1, i_2 \in I$,
- $A_{j_1} \cap A_{j_2} = \emptyset$ *for any* $j_1, j_2 \in J$,
- $r : \bigcup_{i \in I} B_i \times \bigcup_{j \in J} A_j \longrightarrow L$.

Moreover, consider two non-empty sets of L*-fuzzy formal contexts (external formal contexts) notated by*

- $\{\langle B_i, T_i, p_i \rangle : i \in I\}$*, whereby* $\mathcal{C}_i = \langle B_i, T_i, p_i \rangle$,
- $\{\langle O_j, A_j, q_j \rangle : j \in J\}$*, whereby* $\mathcal{D}_j = \langle O_j, A_j, q_j \rangle$.

A second order formal context is a tuple

$$\Big\langle \bigcup_{i \in I} B_i,\ \{\mathcal{C}_i; i \in I\},\ \bigcup_{j \in J} A_j,\ \{\mathcal{D}_j; j \in J\},\ \bigcup_{(i,j) \in I \times J} r_{i,j} \Big\rangle,$$

whereby $r_{i,j} : B_i \times A_j \longrightarrow L$ *is defined as* $r_{i,j}(o, a) = r(o, a)$ *for any* $o \in B_i$ *and* $a \in A_j$.

The Chu construction [8] is a theoretical process that, from a symmetric monoidal closed (autonomous) category and a dualizing object, generates a *-autonomous category. The basic theory of *-autonomous categories and their properties are given in [5, 6].

In the following, the construction will be applied on ChuCors and the dualizing object $\perp = \langle \{\diamond\}, \{\diamond\}, \neq \rangle$ as inputs. In this section it is shown how second order FCA [14] is connected to the output of such construction.

The category generated by the Chu construction and ChuCors and $\perp$ will be denoted by CHU(ChuCors, $\perp$):

- Its objects are triplets of the form $\langle \mathcal{C}, \mathcal{D}, \rho \rangle$ where
  - $\mathcal{C}$ and $\mathcal{D}$ are objects of the input category ChuCors (i.e. formal contexts)
  - ρ is an arrow in $\mathrm{Chu}(\mathcal{C} \boxtimes \mathcal{D}, \bot)$
- Its morphisms are pairs of the form $\langle \varphi, \psi \rangle \colon \langle \mathcal{C}_1, \mathcal{C}_2, \rho_1 \rangle \to \langle \mathcal{D}_1, \mathcal{D}_2, \rho_2 \rangle$ where $\mathcal{C}_i$ and $\mathcal{D}_i$ are formal contexts for $i \in \{1, 2\}$ and
  - φ and ψ are elements from $\mathrm{Chu}(\mathcal{C}_1, \mathcal{D}_1)$ and $\mathrm{Chu}(\mathcal{D}_2, \mathcal{C}_2)$, respectively, such that the following diagram commutes

$$\begin{array}{ccc} \mathcal{C}_1 \boxtimes \mathcal{D}_2 & \xrightarrow{\mathcal{C}_1 \boxtimes \psi} & \mathcal{C}_1 \boxtimes \mathcal{C}_2 \\ {\scriptstyle \varphi \boxtimes \mathcal{D}_2}\downarrow & & \downarrow{\scriptstyle \rho_1} \\ \mathcal{D}_1 \boxtimes \mathcal{D}_2 & \xrightarrow[\rho_2]{} & \bot \end{array}$$

  or, equivalently, the following equality holds

$$(\mathcal{C}_1 \boxtimes \psi) \circ \rho_1 = (\varphi \boxtimes \mathcal{D}_2) \circ \rho_2$$

There are some interesting facts in the previous construction with respect to the second order FCA [14]:

1. To begin with, every object $\langle \mathcal{C}_1, \mathcal{C}_2, \rho \rangle$ in $\mathrm{CHU}(\mathrm{ChuCors}_{\mathcal{L}}, \bot)$, and recall that $\rho \in \mathrm{Chu}(\mathcal{C}_1 \boxtimes \mathcal{C}_2, \bot)$, can be represented as a second order formal context (from Definition 11). Simply take into account that, from basic properties of the tensor product, we can obtain $\mathrm{Chu}(\mathcal{C}_1 \boxtimes \mathcal{C}_2, \bot) \cong \mathrm{Chu}(\mathcal{C}_1, \mathcal{C}_2^*)$.
   Specifically, as ChuCors is a closed monoidal category, we have that for every three formal contexts $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ the following isomorphism holds

$$\mathrm{ChuCors}(\mathcal{C}_1 \boxtimes \mathcal{C}_2, \mathcal{C}_3) \cong \mathrm{ChuCors}(\mathcal{C}_1, \mathcal{C}_2 \multimap \mathcal{C}_3),$$

   whereby $\mathcal{C}_2 \multimap \mathcal{C}_3$ denotes the value at $\mathcal{C}_3$ of the right adjoint and recall that $\mathcal{C}_2 \multimap \bot \cong \mathcal{C}_2^*$ because ChuCors is *-autonomous. The other necessary details about closed monoidal categories and the corresponding notations one can find in [6].
2. Similarly, any second order formal context (from Definition 11) is representable by an object of $\mathrm{CHU}(\mathrm{ChuCors}_{\mathcal{C}}, \bot)$.

6 Conclusions and future work

After introducing the basic definitions needed from category theory and formal concept analysis, in this paper we have studied two different product constructions in the category ChuCors, namely the categorical product and the tensor product. The existence of products allows to represent tables and, hence, binary relations; the tensor product is proved to fulfill the required properties of a bifunctor, which enables us to consider the Chu construction on the category ChuCors. As a first application, we have sketched the representation of

second order formal concept analysis [14] in terms of the Chu construction on the category ChuCors.

The use of different subcategories of ChuCors as input to the Chu construction seems to be an interesting way of obtaining different existing generalizations of FCA. For future work, we are planning to provide representations based on the Chu construction for one-sided FCA, heterogeneous FCA, multi-adjoint FCA, etcetera.

References

1. S. Abramsky. Coalgebras, Chu Spaces, and Representations of Physical Systems. *Journal of Philosophical Logic*, 42(3):551–574, 2013.
2. S. Abramsky. Big Toy Models: Representing Physical Systems As Chu Spaces. *Synthese*, 186(3):697–718, 2012.
3. C. Alcalde, A. Burusco, R. Fuentes-González, The use of two relations in L-fuzzy contexts. *Information Sciences*, 301:1–12, 2015.
4. L. Antoni, S. Krajči, O. Krídlo, B. Macek, L. Pisková, On heterogeneous formal contexts. *Fuzzy Sets and Systems*, 234:22–33, 2014.
5. M. Barr, **-Autonomous categories*, vol. 752 of Lecture Notes in Mathematics. Springer-Verlag, 1979.
6. M. Barr, Ch. Wells, *Category theory for computing science*, 2nd ed., Prentice Hall International (UK) Ltd., 1995.
7. R. Bělohlávek. Concept lattices and order in fuzzy logic. *Annals of Pure and Applied Logic*, 128:277–298, 2004.
8. P.-H. Chu, Constructing *-autonomous categories. Appendix to [5], pages 103–107.
9. J. T. Denniston, A. Melton, and S. E. Rodabaugh. Formal concept analysis and lattice-valued Chu systems. *Fuzzy Sets and Systems*, 216:52–90, 2013.
10. P. Hitzler and G.-Q. Zhang. A cartesian closed category of approximable concept structures. *Lecture Notes in Computer Science*, 3127:170–185, 2004.
11. M. Huang, Q. Li, and L. Guo. Formal Contexts for Algebraic Domains. *Electronic Notes in Theoretical Computer Science*, 301:79–90, 2014.
12. S. Krajči. A categorical view at generalized concept lattices. *Kybernetika*, 43(2):255–264, 2007.
13. O. Krídlo, S. Krajči, and M. Ojeda-Aciego. The category of L-Chu correspondences and the structure of L-bonds. *Fundamenta Informaticae*, 115(4):297–325, 2012.
14. O. Krídlo, P. Mihalčin, S. Krajči, and L. Antoni. Formal concept analysis of higher order. *Proceedings of Concept Lattices and their Applications (CLA)*, 117–128, 2013.
15. O. Krídlo and M. Ojeda-Aciego. On L-fuzzy Chu correspondences. *Intl J of Computer Mathematics*, 88(9):1808–1818, 2011.
16. O. Krídlo and M. Ojeda-Aciego. Revising the link between L-Chu Correspondences and Completely Lattice L-ordered Sets. *Annals of Mathematics and Artificial Intelligence* 72:91–113, 2014.
17. M. Krötzsch, P. Hitzler, and G.-Q. Zhang. Morphisms in context. *Lecture Notes in Computer Science*, 3596:223–237, 2005.
18. H. Mori. Chu correspondences. *Hokkaido Mathematical Journal*, 37:147–214, 2008.
19. J.G. Stell. Formal Concept Analysis over Graphs and Hypergraphs. *Lecture Notes in Computer Science*, 8323:165–179, 2014.
20. G.-Q. Zhang and G. Shen. Approximable concepts, Chu spaces, and information systems. *Theory and Applications of Categories*, 17(5):80–102, 2006.

From formal concepts to analogical complexes

Laurent Miclet[1] and Jacques Nicolas[2]

[1] Université de Rennes 1, UMR IRISA, Dyliss team, Rennes, France,
miclet@univ-rennes1.fr
[2] Inria Rennes, France, jacques.nicolas@inria.fr

Abstract. Reasoning by analogy is an important component of common sense reasoning whose formalization has undergone recent improvements with the logical and algebraic study of the analogical proportion. The starting point of this study considers analogical proportions on a formal context. We introduce analogical complexes, a companion of formal concepts formed by using analogy between four subsets of objects in place of the initial binary relation. They represent subsets of objects and attributes that share a maximal analogical relation. We show that the set of all complexes can be structured in an analogical complex lattice and give explicit formulae for the computation of their infimum and supremum.

Keywords: analogical reasoning, analogical proportion, formal concept, analogical complex, lattice of analogical complexes

1 Introduction

Analogical reasoning [4] plays an important role in human reasoning. It enables us to draw plausible conclusions by exploiting parallels between situations, and as such has been studied in AI for a long time, e.g., [5, 9] under various approaches [3]. A key pattern which is associated with the idea of analogical reasoning is the notion of analogical proportion (AP), i. e. a statement between two pairs (A, B) and (C, D) of the form 'A is to B as C is to D' where all elements A, B, C, D are in a same category .

However, it is only in the last decade that researchers working in computational linguistics have started to study these proportions in a formal way [6, 17, 19]. More recently, analogical proportions have been shown as being of particular interest for classification tasks [10] or for solving IQ tests [2]. Moreover, in the last five years, there has been a number of works, e.g., [11, 15] studying the propositional logic modeling of analogical proportions.

In all previous cases, the ability to work on the set of all possible analogical proportions is required, either for checking missing objects or attributes or for making informed recommendations or more generally ensuring the completeness and efficiency of reasoning. In practice the analysis of objects composed of binary attributes, such as those studied by Formal Concept Analysis, is an important and easy context where AP are used. The question is whether it is possible to obtain a good representation of the space of all AP by applying the principles of

FCA. A heuristic algorithm to discover such proportions by inspecting a lattice of formal concepts has been proposed in [14]. Moreover, a definition of an analogical proportion between formal concepts has been given in [13], as a particular case of proportions between elements of a lattice, studied also in [18].

In this paper, we are interested in a slightly different task involving a more integrated view of concept categorization and analogy: looking for the structure of the space of all AP. Our goal is to build an extension of formal concepts considering the presence of analogical proportions as the funding relation instead of the initial binary relation between objects and attributes. We call this extension *analogical complexes*, which isolate subcontexts in formal contexts with a certain structure reflecting the existence of a maximal analogical proportion between subsets of objects and subsets of attributes.

2 Basics on Analogical Proportion

Definition 1 (Analogical proportion [7, 12]). *An analogical proportion (AP) on a set X is a quaternary relation on X, i.e. a subset of X^4 whose elements (x, y, z, t), written $x : y :: z : t$, which reads 'x is to y as z is to t', must obey the following two axioms:*

1. *Symmetry of 'as': $x : y :: z : t \Leftrightarrow z : t :: x : y$*
2. *Exchange of means: $x : y :: z : t \Leftrightarrow x : z :: y : t$*

In case of formal contexts, objects are described by boolean attributes. An AP (x, y, z, t) between four Boolean variables exists if the following formula is true:

$$(x \wedge \neg y) \Leftrightarrow (z \wedge \neg t) \text{ and } (y \wedge \neg x) \Leftrightarrow (t \wedge \neg z)$$

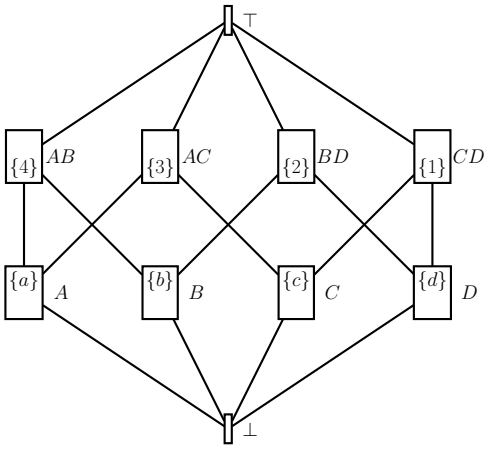
Basically, the formula expresses that the dissimilarity observed between x and y is the same as the dissimilarity between z and t. An equivalent formula is

$$x \neq y \Leftrightarrow (x = z \wedge y = t) \text{ and } x = y \Leftrightarrow z = t$$

It has 6 models of Boolean 4-tuples among the 16 possible ones. Note that this includes the trivial cases where $x = y = z = t$. Since we are only interested in this paper in non trivial analogical proportions, we further require that $x \neq t$ and $y \neq z$. This reduces the number of possible Boolean 4-tuples in AP to four and it leads to the notion of analogical schema that we will use for the definition of analogical complexes.

Definition 2 (Analogical schema). *The binary matrix* $AS = \begin{pmatrix} 0\,0\,1\,1 \\ 0\,1\,0\,1 \\ 1\,0\,1\,0 \\ 1\,1\,0\,0 \end{pmatrix}$ *is called an analogical schema. We write $AS(i, j)$ if the value at row i and column j of matrix AS is 1 (e.g. AS(1,3) and AS(1,4)) .*

The analogical schema may be seen as a formal context on four objects o_1, o_2, o_3, o_4 that are in the non-trivial AP: $o_1 : o_2 :: o_3 : o_4$. The figure 1 shows the associated concept lattice. In this lattice, $A \wedge D = B \wedge C$ and $A \vee D = B \vee C$. The figure also give names for each column and row profiles that we call object and attribute types: for instance the first column as type 1 and the second row as type b.



	1	2	3	4
a			×	×
b		×		×
c	×		×	
d	×	×		

Fig. 1. Left:Concept lattice of an analogical schema (reduced labeling). Analogical schema with object and attribute types.

We use in this paper the zoo dataset proposed by R. Forsyth [8] for illustration purpose. We call smallzoo the formal context extracted from this database corresponding to attributes 2 to 9 and to the objects corresponding to the two largest classes 1 and 2. Moreover, this context has been clarified and we have chosen arbitrarily one object for each of the 10 different types of objects with different attribute profiles. The corresponding table is given below.

smallzoo	2	3	4	5	6	7	8	9	18
	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	type
1 aardvark	1	0	0	1	0	0	1	1	1
12 chicken	0	1	1	0	1	0	0	0	2
17 crow	0	1	1	0	1	0	1	0	2
20 dolphin	0	0	0	1	0	1	1	1	1
22 duck	0	1	1	0	1	1	0	0	2
28 fruitbat	1	0	0	1	1	0	0	1	1
42 kiwi	0	1	1	0	0	0	1	0	2
49 mink	1	0	0	1	0	1	1	1	1
59 penguin	0	1	1	0	0	1	1	0	2
64 platypus	1	0	1	1	0	1	1	0	1

The formal concept lattice is provided in figure 2, as computed by FCA Extension [16]. It contains 31 elements. The central elements (at least two objects and two attributes) are listed below:

$c(3)$	$(\{20;49;59;64\},\{7;8\})$	$c(6)$	$(\{1;20;28;49\},\{5;9\})$
$c(7)$	$(\{1;20;49;64\},\{5;8\})$	$c(8)$	$(\{1;20;49\},\{5;8;9\})$
$c(9)$	$(\{20;49;64\},\{5;7;8\})$	$c(10)$	$(\{20;49\},\{5;7;8;9\})$
$c(12)$	$(\{17;42;59;64\},\{4;8\})$	$c(13)$	$(\{22;59;64\},\{4;7\})$
$c(14)$	$(\{59;64\},\{4;7;8\})$	$c(15)$	$(\{12;17;22;42;59\},\{3;4\})$
$c(16)$	$(\{17;42;59\},\{3;4;8\})$	$c(17)$	$(\{22;59\},\{3;4;7\})$
$c(19)$	$(\{12;17;22\},\{3;4;6\})$	$c(22)$	$(\{1;28;49;64\},\{2;5\})$
$c(23)$	$(\{1;28;49\},\{2;5;9\})$	$c(24)$	$(\{1;49;64\},\{2;5;8\})$
$c(25)$	$(\{1;49\},\{2;5;8;9\})$	$c(26)$	$(\{49;64\},\{2;5;7;8\})$

Example 1. If one extracts in smallzoo the subcontext crossing (12, 28, 59, 49) - that is, (chicken, fruitbat, penguin, mink)- and (7, 2, 3, 6) -(aquatic, hair, feathers, airborne)-, it is clearly an analogical schema.

The 4-tuple (chicken : fruitbat :: penguin : mink) is an analogical proportion that finds a support using attributes (aquatic, hair, feathers, airborne). Each attribute reflects one of the four possible types of Boolean analogy. For instance, hair is false for chicken and penguin and true for fruitbat and mink whereas feathers is true for chicken and penguin and false for fruitbat and mink. The observed analogy can be explained thanks to this typology: the dissimilarity between chicken and fruitbat based on the opposition feather/hair is the same as the dissimilarity between penguin and mink and there are two other opposite attributes, airborne and aquatic, that explain the similarity within each 'is to' relation. Note that the analogical schema if fully symmetric and thus one could also in principle write AP between attributes: hair:feathers::aquatic: airborne.

3 An analogical complex is to an analogical proportion as a concept is to a binary relation

3.1 Analogical complexes

A formal concept on a context (X, Y, I) is a maximal subcontext for which relation I is valid. We define analogical complexes in the same way: they are maximal subcontexts for which the 4-tuples are in AP. This requires to split objects and attributes in four classes.

Definition 3 (Analogical complex). *Given a formal context (X, Y, I), a set of objects $O \subseteq X$, $O = O_1 \cup O_2 \cup O_3 \cup O_4$, a set of attributes $A \subseteq Y$, $A = A_1 \cup A_2 \cup A_3 \cup A_4$, and a binary relation I, the subcontext (O, A) forms an analogical complex $(O_{1,4}, A_{1,4})$ iff*

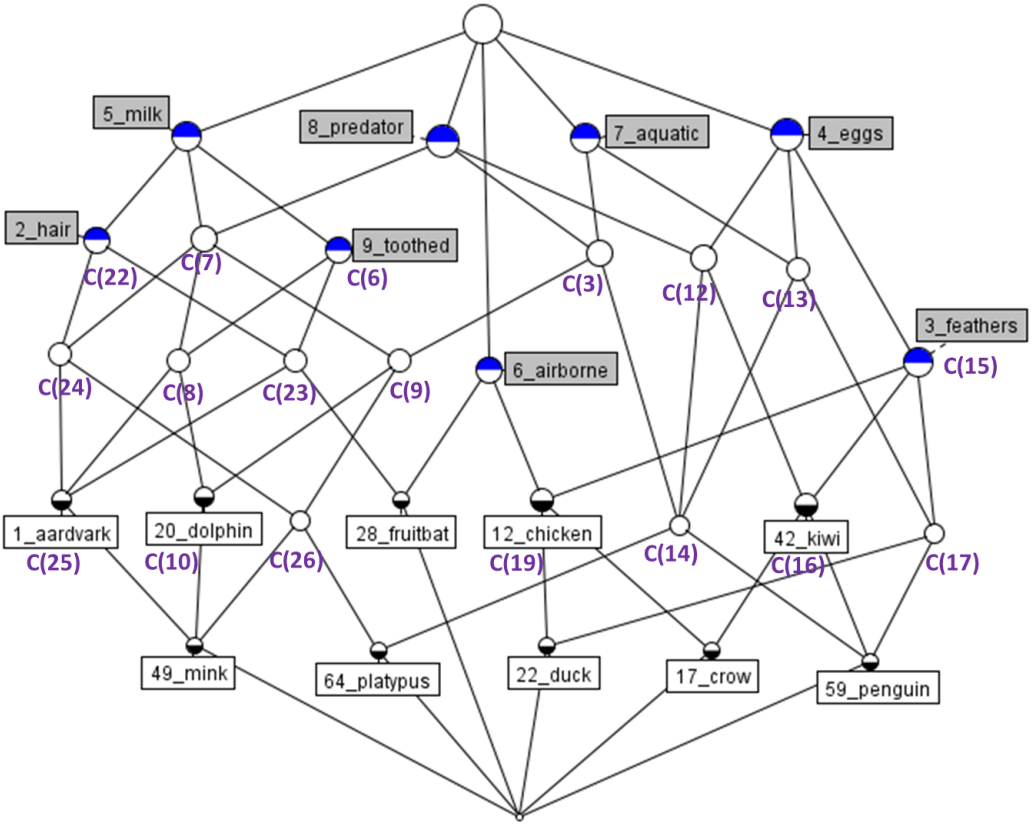


Fig. 2. Formal concept lattice of formal context smallzoo. Drawing from Concept Explorer [20].

1. *The binary relation is compatible with the analogical schema AS:*
   $\forall o \in O_i, i = 1..4,\ \forall a \in A_j, j = 1..4,\ I(o,a) \Leftrightarrow AS(i,j)$.
2. *The context is maximal with respect to the first property ($\oplus$ denotes the exclusive or and $\setminus$ the set-theoretic difference):*
   $\forall o \in X \setminus O,\ \exists j \in [1,4], \exists a \in A_j, I(o,a) \oplus AS(i,j)$.
   $\forall a \in Y \setminus A,\ \exists i \in [1,4], \exists o \in O_i, I(o,a) \oplus AS(i,j)$.

The first property states that the value of an attribute for an object in a complex is a function of object type and attribute type (integer from 1 to 4) given by the analogical schema. The second property states that adding an object (resp. an attribute) to the complex would discard the first property for at least one attribute (resp. object) value. Note that the ways analogical schema or analogical complex are defined are completely symmetric. Thus the role of objects and attributes may be interchanged in all properties on analogical complexes.

Example 2. We extract two subcontexts from smallzoo, highlighting analogical schemas by sorting rows and columns.

	A_1		A_2			A_3		A_4
	a_7	a_8	a_2	a_5	a_9	a_3	a_4	a_6
O_1 $o_{12}(chicken)$	0	0	0	0	0	1	1	1
O_2 $o_{28}(fruitbat)$	0	0	1	1	1	0	0	1
O_3 $o_{59}(penguin)$	1	1	0	0	0	1	1	0
O_4 $o_{49}(mink)$	1	1	1	1	1	0	0	0

	A_1	A_2	A_3	A_4
	a_7			a_6
O_1 o_{12}	0			1
o_{17}	0			1
o_{28}	0			1
O_2 o_{12}	0			1
o_{17}	0			1
o_{28}	0			1
O_3 o_{20}	1			0
o_{49}	1			0
o_{59}	1			0
o_{64}	1			0
O_4 o_{20}	1			0
o_{49}	1			0
o_{59}	1			0
o_{64}	1			0

These subcontexts are maximal in the sense that it is not possible to add an object or an attribute without breaking the analogical proportion. They are associated to the following analogical complexes:

$$\big((\{12\},\{28\},\{59\},\{49\}),\ (\{7,8\},\{2,5,9\},\{3,4\},\{6\})\big)$$

$$\big((\{12,17,28\},\{12,17,28\},\{20,49,59,64\},\{20,49,59,64\}),\ (\{7\},\emptyset,\emptyset,\{6\})\big)$$

The first example provides a strong analogical relation between four animals in the context smallzoo since it uses all attributes and all the types of analogy. Attribute clusters correspond to aquatic predators, toothed animals with hair and milk, birds (feathers and eggs) and flying animals (airborne). The second example shows some of the sets in analogical complexes can be empty. In such a case some sets may be duplicated. Among all complexes, those that exhibit all types of analogy are particularly meaningful: we call them complete complexes.

3.2 Complete analogical complexes (CAC)

Definition 4. *A complex* $\mathcal{C} = (O_{1,4}, A_{1,4})$ *is* complete *if none of its eight sets are empty.*

By construction, if $\mathcal{CA} = (O_{1,4}, A_{1,4})$ is a complete analogical complex and if $\mathcal{A} = \bigcup_{i=1,4} A_i$, the following formula holds:

$$\forall (o_1,o_2,o_3,o_4) \in O_{1,4}, \forall (a_1,a_2,a_3,a_4) \in A_{1,4}$$
$$(o_1^{\uparrow} \cap o_4^{\uparrow}) \cap \mathcal{A} = (o_2^{\uparrow} \cap o_3^{\uparrow}) \cap \mathcal{A} = \emptyset \text{ and } o_1^{\uparrow} \cup o_4^{\uparrow} = o_2^{\uparrow} \cap o_3^{\uparrow} = \mathcal{A}$$

The next proposition shows that CAC exhibits strong discrimination and similarity properties among pairs of objects and attributes. The similarity condition alone would lead to the concatenation of independent (non overlapping) formal concepts. The discrimination condition tempers this tendency by requiring the simultaneous presence of opposite pairs.

Proposition 1. *Let us define on a formal context* $FC = (X, Y, I)$ *the relations:*

$$discrimination(o_i, o_j, a_k, a_l) = I(o_i, a_k) \wedge I(o_j, a_l) \wedge \neg I(o_i, a_l) \wedge \neg I(o_j, a_k).$$

$$similarity(o_i, o_j, a_k, a_l) = I(o_i, a_k) \wedge I(o_j, a_k) \wedge I(o_i, a_l) \wedge I(o_j, a_l).$$

A complete analogical complex $(O_{1,4}, A_{1,4})$ *in* FC *corresponds to a maximal subcontext such that:*

1. *object pair discrimination (resp. similarity):* $\forall (o_i, o_j) \in O_i \times O_j$, $i \neq j$, $\exists (a_k, a_l) \in A_k \times A_l$ *such that discrimination*(o_i, o_j, a_k, a_l) *(resp. similarity*(o_i, o_j, a_k, a_l)*);*
2. *attribute pair discrimination (res. similarity):* $\forall (a_k, a_l) \in A_k \times A_l$, $k \neq l$, $\exists (o_i, o_j) \in O_i \times O_j$ *such that discrimination*(o_i, o_j, a_k, a_l) *(resp. similarity*(o_i, o_j, a_k, a_l)*).*

Proof. Since objects and attribute have a completely symmetrical role, it is sufficient to prove the proposition for object pairs. It proceeds easily by enumerating the possible type pairs with different elements. If objects have type 1 and 2 or 3 and 4, attributes allowing object pair discrimination have type b and c and attributes allowing object pair similarity have type a and d. If objects have type 1 and 3 or 2 and 4, attributes allowing object pair discrimination have type a and d and attributes allowing object pair similarity have type b and c. If objects have type 1 and 4 and if $t_1 \in T_1 = \{a, b\}$ and $t_2 \in T_2 = \{c, d\}$, attributes allowing object pair discrimination have type t_1 and t_2 and attributes allowing object pair similarity have different types both in T_1 or both in T_2. If objects have type 2 and 3 and if $t_1 \in T_1 = \{a, c\}$ and $t_2 \in T_2 = \{b, d\}$, attributes allowing object pair discrimination have type t_1 and t_2 and attributes allowing object pair similarity have different types both in T_1 or both in T_2. □

In case of incomplete complexes, some of these properties are no more relevant and a degenerate behaviour may appear: some of the sets may be identical. This fact allows to establish a new proposition on complete complexes:

Proposition 2. *In a complete analogical complex, side-by-side intersections of sets are empty.*

Proof. This property holds since when the intersection of two object (resp. attribute) sets in an analogical complex AC is not empty, then AC contains at least two empty attribute (resp. object) sets. This fact is a consequence of property 1.
Indeed, if an object belongs to two different types, their profiles must be the

same. The discrimination property ensures that the profile of two different object types differ by at least two different attribute with different types (e.g. if the object has type 1 and 3, attributes of type b and c should have different values). Thus it cannot exists attributes of the discriminant type (e.g. attributes of type b and c in the previous case) and the corresponding sets are empty. This completes the proof.

The converse of the proposition is not true: if all side-by-side intersections of sets differ, the complex is not necessary complete. For instance, consider the following context:

	a_1	a_2	a_3	a_4	a_5	a_6
o_1	0	0	0	1	1	1
o_2	0	1	1	1	1	1
o_3	1	0	0	0	0	1
o_4	1	1	1	0	0	0
o_5	1	0	1	1	0	0

It contains the following not complete complex:

$$\big((\{o_1\},\{o_2\},\{o_3\},\{o_4\}),\ (\{a_1\},\{a_2,a_3\},\emptyset,\{a_4,a_5\})\big)$$

4 The lattice of analogical complexes

Definition 5 (Partial Order on analogical complexes). *Given two analogical complexes $\mathcal{C}^1 = (O^1_{1,4}, A^1_{1,4})$ and $\mathcal{C}^2 = (O^2_{1,4}, A^2_{1,4})$, the partial order $\leq$ is defined by*

$$\mathcal{C}^1 \leq \mathcal{C}^2 \quad \textit{iff} \quad \big(O^1_i \subseteq O^2_i \textit{ for } i = 1,4 \quad \textit{and} \quad A^2_i \subseteq A^1_i \textit{ for } i = 1,4\big).$$

$\mathcal{C}^1$ is called a sub-complex of $\mathcal{C}^2$ and $\mathcal{C}^2$ is called a super-complex of $\mathcal{C}^1$

As for formal concepts, the set of all complexes has a lattice structure. Let us first define a derivation operator on analogical quadruplets:

Definition 6 (Derivation on set quadruplets).

Let $O = O_1 \cup O_2 \cup O_3 \cup O_4$ be a set of objects partitioned in four subsets, and A be a set of attributes. For all i and $j \in [1,4]$, one defines $O_i^{'j} = \{a \in A \mid \forall o \in O_i\ I(o,a) \Leftrightarrow AS(i,j)\}$

Let $A = A_1 \cup A_2 \cup A_3 \cup A_4$ be a set of attributes partitioned in four subsets, and O be a set of objects. For all i and $j \in [1,4]$, one defines $A_i^{'j} = \{o \in O \mid \forall a \in A_i\ I(o,a) \Leftrightarrow AS(i,j)\}$

Finally, we define the derivation on quadruplets as follows:

$$O'_{1,4} = (\bigcap_{j=1}^{4} O_j^{'1}, \bigcap_{j=1}^{4} O_j^{'2}, \bigcap_{j=1}^{4} O_j^{'3}, \bigcap_{j=1}^{4} O_j^{'4})$$

$$A'_{1,4} = (\bigcap_{j=1}^{4} A_j^{'1}, \bigcap_{j=1}^{4} A_j^{'2}, \bigcap_{j=1}^{4} A_j^{'3}, \bigcap_{j=1}^{4} A_j^{'4})$$

Example 3. Consider $O = (\{12\}, \{28\}, \{59\}, \{49\})$. One has: $O_1'^1 = \{a \in A \mid \neg I(12, a)\} = \{2, 5, 7, 8, 9\}$;
$O_2'^1 = \{a \in A \mid \neg I(28, a)\} = \{3, 4, 7, 8\}$;
$O_3'^1 = \{a \in A \mid I(59, a)\} = \{3, 4, 7, 8\}$;
$O_4'^1 = \{a \in A \mid I(49, a)\} = \{2, 4, 5, 7, 8\}$ Then $O_1' = \bigcap_{j=1}^{4} O_j'^1 = \{7, 8\}$
Finally, $O' = (\{7, 8\}, \{2, 5, 9\}, \{3, 4\}, \{6\})$.

We exhibit a basic theorem for these complexes that naturally extends the basic theorem on concepts:

Proposition 3. *Given two analogical complexes* $\mathcal{C}^1 = (O^1_{1,4}, A^1_{1,4})$ *and* $\mathcal{C}^2 = (O^2_{1,4}, A^2_{1,4})$,

- *The join of* $\mathcal{C}^1$ *and* $\mathcal{C}^2$ *is defined by* $\mathcal{C}^1 \wedge \mathcal{C}^2 = (\mathcal{O}_{1,4}, \mathcal{A}_{1,4})$ *where*

$$\forall i \in [1, 4]\ \mathcal{O}_i = O_i(\mathcal{C}^1) \cap O_i(\mathcal{C}^2)$$

$$\mathcal{A}_{1,4} = \big(A_1(\mathcal{C}^1) \cup A_1(\mathcal{C}^2), A_2(\mathcal{C}^1) \cup A_2(\mathcal{C}^2), A_3(\mathcal{C}^1) \cup A_3(\mathcal{C}^2), A_4(\mathcal{C}^1) \cup A_4(\mathcal{C}^2)\big)''$$

- *The meet of* $\mathcal{C}^1$ *and* $\mathcal{C}^2$ *is defined by* $\mathcal{C}^1 \vee \mathcal{C}^2 = (\mathcal{O}_{1,4}, \mathcal{A}_{1,4})$ *where*

$$\mathcal{O}_{1,4} = \big(O_1(\mathcal{C}^1) \cup O_1(\mathcal{C}^2), O_2(\mathcal{C}^1) \cup O_2(\mathcal{C}^2), O_3(\mathcal{C}^1) \cup O_3(\mathcal{C}^2), O_4(\mathcal{C}^1) \cup O_4(\mathcal{C}^2)\big)''$$

Proof. The meet and the join are dual and one only needs to prove the proposition for the join. The ordering by set inclusion requires the set of objects $\mathcal{O}_i$ of $\mathcal{C}^1 \wedge \mathcal{C}^2$ to be included in $O_i(\mathcal{C}^1) \cap O_i(\mathcal{C}^2)$ and its set of attributes $\mathcal{A}_j$ to be included in $A_j(\mathcal{C}^1) \cup A_j\mathcal{C}^2)$. Taking exactly the intersection of objects thus ensures the set of objects to be maximal. The corresponding maximal sets of attributes may be inferred using the derivation operator ' we have just defined. Another way to generate these sets is to apply the derivation operator twice on the union of sets of attributes.

Example 4. The complex lattice of smallzoo has 24 elements, including 18 complete complexes. It is sketched in figure 3.

In this lattice, for example, the join of the analogical complex numbered 9 and 12, which are as follows

$$9 = \big((\{12\}, \{28\}, \{59, 64\}, \{20, 49\}),\ (\{7\}, \{9\}, \{4\}, \{6\})\big)$$

$$12 = \big((\{12, 17\}, \{28\}, \{59\}, \{49, 64\}),\ (\{7\}, \{2, 5\}, \{3\}, \{6\})\big)$$

is number 15, namely:

$$15 = \big((\{12\}, \{28\}, \{59\}, \{49\}),\ (\{7, 8\}, \{2, 5, 9\}, \{3, 4\}, \{6\})\big)$$

The resulting object sets are for each type the intersection of the two joined object sets. The resulting attribute sets contain for each type the union of the two joined attribute sets and may contain other elements with a correct profile on all objects. For instance, $A_1(9 \wedge 12) = \{7, 8\}$ is made of the union of $A_1(9)$

and $A_1(12)$ ($\{7\}$) plus attribute 8 since 8 has the right profile $(0, 0, 1, 1)$ on $O_{1,4}$ (that is, $\neg I(12, 8), \neg I(28, 8)$, $I(59, 8)$ and $I(49, 8)$).

The meet of the analogical complexes numbered 9 and 12 is number 19, namely

$$19 = \big((\{12, 17, 28\}, \{12, 17, 28\}, \{20, 49, 59, 64\}, \{20, 49, 59, 64\}),\ (\{7\}, \emptyset, \emptyset, \{6\})\big)$$

5 Conclusion

We have introduced a new conceptual object called analogical complex that uses a complex relation, analogical proportion, to compare objects with respect to their attribute values. Although this relation works on set quadruplets instead of simple sets like in formal concepts, we have shown that it is possible to keep the main properties of concepts, that is, maximality and comparison at the level of object or attribute pairs. The set of all complexes are structured within a lattice that contains two types of elements. The most meaningful ones only contain non empty sets and are a strong support for doing analogical inference. An interesting extension of this work would be to develop this inference process for analogical data mining in a way close to rule generation in FCA.

The degenerate case where some of the sets are empty is more frequent than in FCA where their presence is limited to the top or bottom of the lattice. The presence of a single empty set may reflect the lack of some object or attribute and is thus a possible new research direction for completing a knowledge base or an ontology. Particularly, analogy in a Boolean framework introduces a form of negation through the search of dissimilarities (discrimination) between objects.

We have written an implementation the search for complete analogical complexes, using the Answer Set Programming framework [1]. The properties of definition 3 are translated straightforwardly in logical constraints and the search of all complexes is achieved by an ASP solver looking for all solutions. The description of the ASP program would be beyond the scope of this paper but it can be seen as a relatively simple exercise of extension of the search for formal concepts by adding a few logical constraints. It is likely that most of the existing tools of FCA could be adapted the same way for analogical complex analysis. This would allow to include both categorization and analogy within common data mining environments.

References

1. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM 54(12), 92–103 (Dec 2011), http://doi.acm.org/10.1145/2043174.2043195
2. Correa, W., Prade, H., Richard, G.: When intelligence is just a matter of copying. In: et al., L.D.R. (ed.) Proc. 20th Europ. Conf. on Artificial Intelligence, Montpellier, Aug. 27-31. pp. 276–281. IOS Press (2012)

3. French, R.M.: The computational modeling of analogy-making. Trends in Cognitive Sciences 6(5), 200 – 205 (2002)
4. Gentner, D., Holyoak, K.J., Kokinov, B.N.: The Analogical Mind: Perspectives from Cognitive Science. Cognitive Science, and Philosophy, MIT Press, Cambridge, MA (2001)
5. Hofstadter, D., Mitchell, M.: The Copycat project: A model of mental fluidity and analogy-making. In: Hofstadter, D., The Fluid Analogies Research Group (eds.) Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought. pp. 205–267. Basic Books, Inc., New York, NY (1995)
6. Lepage, Y.: Analogy and formal languages. Electr. Notes Theor. Comput. Sci. 53 (2001)
7. Lepage, Y.: Analogy and formal languages. In: Proc. FG/MOL 2001. pp. 373–378 (2001), (see also http://www.slt.atr.co.jp/ lepage/pdf/dhdryl.pdf.gz)
8. Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
9. Melis, E., Veloso, M.: Analogy in problem solving. In: Handbook of Practical Reasoning: Computational and Theoretical Aspects. Oxford Univ. Press (1998)
10. Miclet, L., Bayoudh, S., Delhay, A.: Analogical dissimilarity: definition, algorithms and two experiments in machine learning. JAIR, 32 pp. 793–824 (2008)
11. Miclet, L., Prade, H.: Handling analogical proportions in classical logic and fuzzy logics settings. Proc. 10th Eur. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09),Verona (2009)
12. Miclet, L., Prade, H.: Handling analogical proportions in classical logic and fuzzy logics settings. In: Proc. 10th Eur. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09),Verona. pp. 638–650. Springer, LNCS 5590 (2009)
13. Miclet, L., Barbot, N., Prade, H.: From analogical proportions in lattices to proportional analogies in formal concepts. In: ECAI - 21th European Conference on Artificial Intelligence. Prague, Czech Republic (Aug 2014)
14. Miclet, L., Prade, H., Guennec, D.: Looking for Analogical Proportions in a Formal Concept Analysis Setting. In: Amedeo Napoli, V.V. (ed.) Conference on Concept Lattices and Their Applications. pp. 295–307. Nancy, France (Oct 2011)
15. Prade, H., Richard, G.: Homogeneous logical proportions: Their uniqueness and their role in similarity-based prediction. Proc. of the 13th International Conference on Principles of Knowledge Representation and Reasoning KR2012 pp. 402 – 412 (2012)
16. Radvansky, M., Sklenar, V.: Fca extension for ms excel 2007, http://www.fca.radvansky.net (2010)
17. Stroppa, N., Yvon, F.: An analogical learner for morphological analysis. In: Online Proc. 9th Conf. Comput. Natural Language Learning (CoNLL-2005). pp. 120–127 (2005)
18. Stroppa, N., Yvon, F.: Analogical learning and formal proportions: Definitions and methodological issues. ENST Paris report (2005)
19. Stroppa, N., Yvon, F.: Du quatrième de proportion comme principe inductif : une proposition et son application à l'apprentissage de la morphologie. Traitement Automatique des Langues 47(2), 1–27 (2006)
20. Yevtushenko, S.: System of data analysis "concept explorer". (in russian). In: Proc. of the 7th national conference on artificial intelligence (KII-2000) ,Russia. pp. 127–134 (2000)

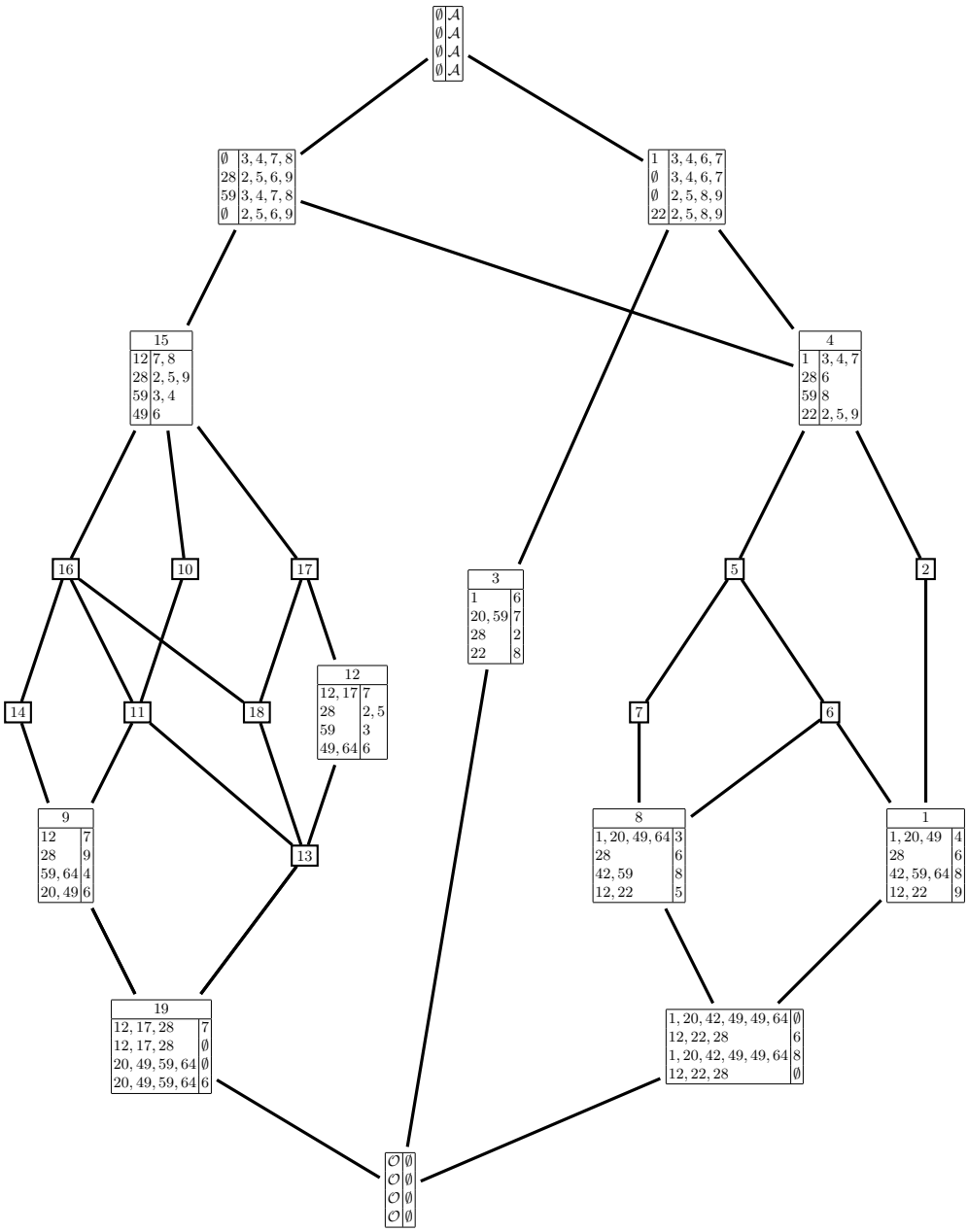


Fig. 3. Hasse diagram of the analogical complex lattice for formal context smallzoo. For reasons of space some nodes are not explicitely given.

Pattern Structures and Their Morphisms

Lars Lumpe[1] and Stefan E. Schmidt[2]

Institut für Algebra,
Technische Universität Dresden
larslumpe@gmail.com[1], midt1@msn.com[2]

Abstract. Projections of pattern structures don't always lead to pattern structures, however residual projections and o-projections do. As a unifying approach, we introduce the notion of pattern morphisms between pattern structures and provide a general sufficient condition for a homomorphic image of a pattern structure being again a pattern structure. In particular, we receive a better understanding of the theory of o-projections.

1 Introduction

Pattern structures within the framework of formal concept analysis have been introduced in [3]. Since then they have turned out to be a useful tool for analysing various real-world applications (cf. [3–7]). In this work we want to point out that the theoretical foundations of pattern structures encourage still some fruitful discussions. In particular, the role projections play within pattern structures for information reduction still needs some further investigation.

The goal of our paper is to establish an adequate concept of pattern morphism between pattern structures, which also gives a better understanding of the concept of o-projections as recently introduced and investigated in [2]. In [8], we showed that projections of pattern structures do not necessarily lead to pattern structures again, however, residual projections do. It turns out that the concept of residual maps between the posets of patterns (w.r.t. two pattern structures) gives the key for a unifying view of o-projections and residual projections.

We also derive that a pattern morphism from a pattern structure to a pattern setup (introduced in this paper), which is surjective on the sets of objects, yields again a pattern structure.

Our main result states that a pattern morphism always induces an adjunction between the corresponding concept lattices. In case the underlying map between the sets of objects is surjective, the induced residuated map between the concept lattices turns out to be surjective too.

The fundamental order theoretic concepts of our paper are nicely presented in the book on *Residuation Theory* by T.S. Blythe and M.F. Janowitz (cf. [1]).

2 Preliminaries

Definition 1 (Adjunction)**.** *Let $\mathbb{P} = (P, \leqslant)$ and $\mathbb{L} = (L, \leqslant)$ be posets; furthermore let $f : P \to L$ and $g : L \to P$ be maps.*

(1) The pair (f, g) is an **adjunction** *w.r.t. $(\mathbb{P}, \mathbb{L})$ if $fx \leqslant y$ is equivalent to $x \leqslant gy$ for all $x \in P$ and $y \in L$. In this case, we will refer to $(\mathbb{P}, \mathbb{L}, f, g)$ as a* **poset adjunction**.

(2) f is **residuated** *from $\mathbb{P}$ to $\mathbb{L}$ if the preimage of a principal ideal in $\mathbb{L}$ under f is always a principal ideal in $\mathbb{P}$, that is, for every $y \in L$ there exists $x \in P$ s.t.*

$$f^{-1}\{t \in L \mid t \leqslant y\} = \{s \in P \mid s \leqslant x\}.$$

(3) g is **residual** *from $\mathbb{L}$ to $\mathbb{P}$ if the preimage of a principal filter in $\mathbb{P}$ under g is always a principal filter in $\mathbb{L}$, that is, for every $x \in P$ there exists $y \in L$ s.t.*

$$g^{-1}\{s \in P \mid x \leqslant s\} = \{t \in L \mid y \leqslant t\}.$$

(4) The dual of $\mathbb{L}$ is given by $\mathbb{L}^{\mathrm{op}} = (L, \geqslant)$ with $\geqslant := \{(x,t) \in L \times L \mid t \leqslant x\}$. The pair (f, g) is a **Galois connection** *w.r.t. $(\mathbb{P}, \mathbb{L})$ if (f, g) is an* **adjunction** *w.r.t. $(\mathbb{P}, \mathbb{L}^{\mathrm{op}})$.*

The following well-known facts are straightforward (cf. [1]).

Proposition 1. *Let $\mathbb{P} = (P, \leqslant)$ and $\mathbb{L} = (L, \leqslant)$ be posets.*

(1) A map $f : P \to L$ is residuated from $\mathbb{P}$ to $\mathbb{L}$ iff there exists a map $g : L \to P$ s.t. (f, g) is an adjunction w.r.t. $(\mathbb{P}, \mathbb{L})$.

(2) A map $g : L \to P$ is residual from $\mathbb{L}$ to $\mathbb{P}$ iff there exists a map $f : P \to L$ s.t. (f, g) is an adjunction w.r.t. $(\mathbb{P}, \mathbb{L})$.

(3) If (f, g) and (h, k) are adjunctions w.r.t. $(\mathbb{P}, \mathbb{L})$ with $f = h$ or $g = k$ then $f = h$ and $g = k$.

(4) If f is a residuated map from $\mathbb{P}$ to $\mathbb{L}$, then there exists a unique residual map f^+ from $\mathbb{L}$ to $\mathbb{P}$ s.t. (f, f^+) is an adjunction w.r.t. $(\mathbb{P}, \mathbb{L})$. In this case, f^+ is called the **residual map** *of f.*

(5) If g is a residual map from $\mathbb{L}$ to $\mathbb{P}$, then there exists a unique residuated map g^- from $\mathbb{P}$ to $\mathbb{L}$ s.t. (g^-, g) is an adjunction w.r.t. $(\mathbb{P}, \mathbb{L})$. In this case, g^- is called the **residuated map** *of g.*

Definition 2. *Let $\mathbb{P} = (P, \leqslant)$ be a poset and $T \subseteq P$. Then*

(1) The **restriction** *of $\mathbb{P}$ onto T is given by $\mathbb{P}|T := (T, \leqslant \cap (T \times T))$, which clearly is a poset too.*

(2) The **canonical embedding** *of $\mathbb{P}|T$ into $\mathbb{P}$ is given by the map $T \to P, t \mapsto t$.*

(3) T is a **kernel system** *in $\mathbb{P}$ if the canonical embedding τ of $\mathbb{P}|T$ into $\mathbb{P}$ is residuated. In this case, the residual map φ of τ will also be called the* **residual map** *of T in $\mathbb{P}$. The composition $\kappa := \tau \circ \varphi$ is referred to as the* **kernel operator** *associated with T in $\mathbb{P}$.*

(4) Dually, T is a **closure system** *in $\mathbb{P}$ if the cannonical embedding τ of $\mathbb{P}|T$ into $\mathbb{P}$ is residual. In this case, the residuated map ψ of τ will also be called the* **residuated map** *of T in $\mathbb{P}$. The composition $\gamma := \tau \circ \psi$ is referred to as the* **closure operator** *associated with T in $\mathbb{P}$.*

(5) *A map* $\kappa : P \to P$ *is a* **kernel operator** *on* $\mathbb{P}$ *if* $s \leqslant x$ *is equivalent to* $s \leqslant \kappa x$ *for all* $s \in \kappa P$ *and* $x \in P$.
Remark: In this case, κP *forms a kernel system in* $\mathbb{P}$*, the kernel operator of which is* κ.

(6) *Dually, a map* $\gamma : P \to P$ *is a* **closure operator** *on* $\mathbb{P}$ *if* $x \leqslant t$ *is equivalent to* $\gamma x \leqslant t$ *for all* $x \in P$ *and* $t \in \gamma P$.
Remark: In this case, φP *forms a closure system in* $\mathbb{P}$*, the closure operator of which is* γ.

The following known facts will be needed for the sequel (cf. [1]) .

Proposition 2. *Let* $\mathbb{P} = (P, \leqslant)$ *and* $\mathbb{L} = (L, \leqslant)$ *be posets.*

(1) *If* f *is a residuated map from* $\mathbb{P}$ *to* $\mathbb{L}$ *then* f *preserves all existing suprema in* $\mathbb{P}$*, that is, if* $s \in P$ *is the supremum (least upper bound) of* $X \subseteq P$ *in* $\mathbb{P}$ *then* fs *is the supremum of* fX *in* $\mathbb{L}$.
In case $\mathbb{P}$ *and* $\mathbb{L}$ *are complete lattices, the reverse holds too: If a map* f *from* $\mathbb{P}$ *to* $\mathbb{L}$ *preserves all suprema, that is,*

$$f(\sup_{\mathbb{P}} X) = \sup_{\mathbb{L}} fX \text{ for all } X \subseteq P,$$

then f is residuated.

(2) *If g is a residual map from* $\mathbb{L}$ *to* $\mathbb{P}$*, then g preserves all existing infima in* $\mathbb{L}$*, that is, if* $t \in L$ *is the infimum (greatest lower bound) of* $Y \subseteq L$ *in* $\mathbb{L}$ *then* gt *is the infimum of* gY *in* $\mathbb{P}$.
In case $\mathbb{P}$ *and* $\mathbb{L}$ *are complete lattices, the reverse holds too: If a map g from* $\mathbb{L}$ *to* $\mathbb{P}$ *preserves all infima, that is,*

$$f(\inf_{\mathbb{P}} Y) = \inf_{\mathbb{L}} gY \text{ for all } Y \subseteq L,$$

then g is residual.

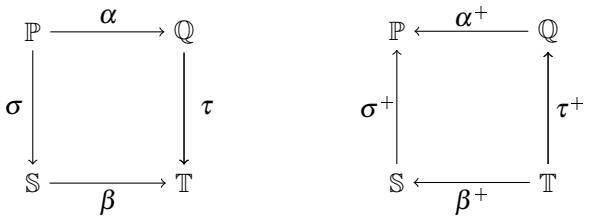
(3) *For an adjunction* (f, g) *w.r.t.* $(\mathbb{P}, \mathbb{L})$ *the following hold:*
- (a1) f *is an isotone map from* $\mathbb{P}$ *to* $\mathbb{L}$.
- (a2) $f \circ g \circ f = f$
- (a3) fP *is a kernel system in* $\mathbb{L}$ *with* $f \circ g$ *as associated kernel operator on* $\mathbb{L}$*. In particular,* $L \to P, y \mapsto fgy$ *is a residual map from* $\mathbb{L}$ *to* $\mathbb{L}|fP$.
- (b1) g *is an isotone map from* $\mathbb{L}$ *to* $\mathbb{P}$.
- (b2) $g \circ f \circ g = g$
- (b3) gL *is a closure system in* $\mathbb{P}$ *with* $g \circ f$ *as associated closure operator on* $\mathbb{P}$*. In particular,* $P \to gL, x \mapsto gfx$ *is a residuated map from* $\mathbb{P}$ *to* $\mathbb{P}|gL$.

3 Adjunctions and Their Concept Posets

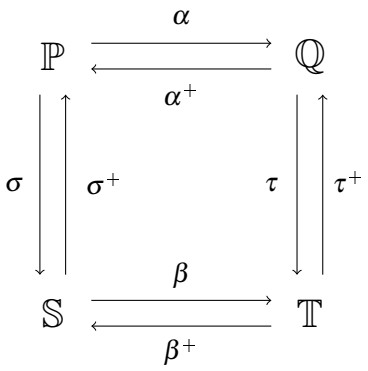
Definition 3. *Let* $\mathcal{P} := (\mathbb{P}, \mathbb{S}, \sigma, \sigma^+)$ *and* $\mathcal{Q} := (\mathbb{Q}, \mathbb{T}, \tau, \tau^+)$ *be poset adjunctions. Then a pair* (α, β) *forms a morphism from* $\mathcal{P}$ *to* $\mathcal{Q}$ *if* $(\mathbb{P}, \mathbb{Q}, \alpha, \alpha^+)$ *and* $(\mathbb{S}, \mathbb{T}, \beta, \beta^+)$ *are poset adjunctions satisfying*

$$\tau \circ \alpha = \beta \circ \sigma$$

Remark: This implies $\alpha^+ \circ \tau^+ = \sigma^+ \circ \beta^+$*, that is, the following diagrams are commutative:*



Next we illustrate the involved poset adjunctions:



Definition 4 (Concept Poset)**.** *For a poset adjunction* $\mathcal{P} = (\mathbb{P}, \mathbb{S}, \sigma, \sigma^+)$ *let*

$$\mathrm{B}\mathcal{P} := \{(p,s) \in P \times S \mid \sigma p = s \wedge \sigma^+ s = p\}$$

denote the set of (**formal**) **concepts** *in* $\mathcal{P}$*. Then the* **concept poset** *of* $\mathcal{P}$ *is given by*

$$\mathbb{B}\mathcal{P} := (\mathbb{P} \times \mathbb{S}) \mid \mathrm{B}\mathcal{P},$$

that is, $(p_0, s_0) \leqslant (p_1, s_1)$ *holds iff* $p_0 \leqslant p_1$ *iff* $s_0 \leqslant s_1$*, for all* $(p_0, s_0), (p_1, s_1) \in B\mathcal{P}$*. If* (p,s) *is a formal concept in* $\mathcal{P}$ *then* p *is referred to as* **extent** *in* $\mathcal{P}$ *and* s *as* **intent** *in* $\mathcal{P}$*.*

Theorem 1. *Let* (α, β) *be a morphism from a poset adjunction* $\mathcal{P} = (\mathbb{P}, \mathbb{S}, \sigma, \sigma^+)$ *to a poset adjunction* $\mathcal{Q} = (\mathbb{Q}, \mathbb{T}, \tau, \tau^+)$*. Then*

$$(\mathbb{B}\mathcal{P}, \mathbb{B}\mathcal{Q}, \Phi, \Psi)$$

is a poset adjunction for

$$\Phi : \mathrm{B}\mathcal{P} \to \mathrm{B}\mathcal{Q}, (p,s) \mapsto (\tau^+ \beta s, \beta s)$$

and

$$\Psi : \mathrm{B}\mathcal{Q} \to \mathrm{B}\mathcal{P}, (q,t) \mapsto (\alpha^+ q, \sigma \alpha^+ q).$$

In addition, if α *is surjective then so is* Φ*.*
Remark: In particular we want to point out that $\alpha^+ q$ *is an extent in* $\mathcal{P}$ *for every extent* q *in* $\mathcal{Q}$ *and similarly,* βs *is an intent in* $\mathcal{Q}$ *for every intent* s *in* $\mathcal{P}$*.*

Proof. Let $(p,s) \in \mathrm{B}\mathcal{P}$ and $(q,t) \in \mathrm{B}\mathcal{Q}$; then $\sigma p = s$ and $\sigma^+ s = p$ and $\tau q = t$ and $\tau^+ t = q$. This implies $\beta s = \beta\sigma p = \tau\alpha p$, thus

$$\Phi(p,s) = (\tau^+\beta s, \beta s) \in \mathrm{B}\mathcal{P}$$

(since $\tau\tau^+\beta s = \tau\tau^+\tau\alpha p = \tau\alpha p = \beta s$). Similarly, $\Psi(q,t) \in \mathrm{B}\mathcal{Q}$.
Assume now that $\Phi(p,s) \leqslant (q,t)$ holds, which implies $\beta s \leqslant t$. It follows that

$$\tau\alpha p = \beta\sigma p = \beta s \leqslant t$$

and hence

$$p \leqslant \alpha^+\tau^+ t = \alpha^+ q,$$

that is, $(p,s) \leqslant \Psi(q,t)$.
Conversely, assume that $(p,s) \leqslant \Psi(q,t)$ holds, which implies $p \leqslant \alpha^+ q$. It follows that

$$p \leqslant \alpha^+ q = \alpha^+\tau^+ t = \sigma^+\beta^+ t,$$

and hence $\beta s = \beta\sigma p \leqslant t$, that is, $\Phi(p,s) \leqslant (q,t)$.
Assume now that α is surjective; then $\alpha \circ \alpha^+ = \mathrm{id}_Q$. Let $(q,t) \in \mathrm{B}\mathcal{P}$, that is, $\tau q = t$ and $\tau^+ t = q$. Then for $p := \alpha^+ q$ and $s := \sigma p$ we have $(p,s) \in \mathrm{B}\mathcal{P}$ since

$$\sigma^+ s = \sigma^+\sigma\alpha^+ q = \sigma^+\sigma\alpha^+\tau^+ t = \sigma^+\sigma\sigma^+\beta^+ t = \sigma^+\beta^+ t = \alpha^+\tau^+ t = \alpha^+ q = p.$$

Our claim is now that $\Phi(p,s) = (q,t)$ holds, that is, $\beta s = t$. The latter is true, since $\alpha p = \alpha\alpha^+ q = q$ implies

$$\beta s = \beta\sigma p = \tau\alpha p = \tau q = t.$$

□

Discussion for clarification: The question was raised whether, in the previous theorem, the residuated map Φ from $\mathrm{B}\mathcal{P}$ to $\mathrm{B}\mathcal{Q}$ allows some modification, since the map

$$P \times S \to Q \times T, (p,s) \mapsto (\alpha p, \beta s)$$

is obviously residuated from $\mathbb{P} \times \mathbb{S}$ to $\mathbb{Q} \times \mathbb{T}$. However, in general the latter map does not restrict to a map from $\mathrm{B}\mathcal{P}$ to $\mathrm{B}\mathcal{Q}$. Indeed, our construction of the map Φ is of the form $(p,s) \mapsto (\alpha' p, \beta s)$. As a warning, we want to point out that, in general, there is no residuated map from $\mathrm{B}\mathcal{P}$ to $\mathrm{B}\mathcal{Q}$ of the form $(p,s) \mapsto (\alpha p, \beta' s)$. The simple reason for this is that βs is an intent in $\mathcal{Q}$ for every intent s in $\mathcal{P}$, while there may exist an extent p in $\mathcal{P}$ such that αp is not an extent in $\mathcal{Q}$.

4 Morphisms between Pattern Structures

Definition 5. *A triple $\mathcal{G} = (G, \mathbb{D}, \delta)$ is a* **pattern setup** *if G is a set, $\mathbb{D} = (D, \sqsubseteq)$ is a poset, and $\delta : G \to D$ is a map. In case every subset of $\delta G := \{\delta g \mid g \in G\}$ has an infimum in $\mathbb{D}$, we will refer to $\mathcal{G}$ as* **pattern structure**. *Then the set*

$$\mathrm{C}_{\mathcal{G}} := \{\inf_{\mathbb{D}} \delta X \mid X \subseteq G\}$$

forms a closure system in $\mathbb{D}$.

If $\mathcal{G} = (G, \mathbb{D}, \delta)$ *and* $\mathcal{H} = (H, \mathbb{E}, \varepsilon)$ *each is a pattern setup, then a pair* (f, φ) *forms a* **pattern morphism** *from* $\mathcal{G}$ *to* $\mathcal{H}$ *if* $f : G \to H$ *is a map and* φ *is a residual map from* $\mathbb{D}$ *to* $\mathbb{E}$ *satisfying* $\varphi \circ \delta = \varepsilon \circ f$, *that is, the following diagram is commutative:*

$$\begin{array}{ccc} G & \xrightarrow{\quad f \quad} & H \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\varepsilon} \\ \mathbb{D} & \xrightarrow[\quad \varphi \quad]{} & \mathbb{E} \end{array}$$

In the sequel we show how our previous considerations apply to pattern structures.

Applications

(1) Let $\mathcal{G}$ be a pattern structure and $\mathcal{H}$ be a pattern setup. If (f, φ) is a pattern morphism from $\mathcal{G}$ to $\mathcal{H}$ with f being surjective, then $\mathcal{H}$ is also a pattern structure.

(2) Let $\mathcal{G} = (G, \mathbb{D}, \delta)$ and $\mathcal{H} = (H, \mathbb{E}, \varepsilon)$ be pattern structures. Also let (f, φ) be a pattern morphism from $\mathcal{G}$ to $\mathcal{H}$.
To apply the previous theorem we give the following construction:
f gives rise to an adjunction (α, α^+) between the power set lattices $\mathbf{2}^G := (2^G, \subseteq)$ and $\mathbf{2}^H := (2^H, \subseteq)$ via

$$\alpha : 2^G \to 2^H, X \mapsto fX$$

and

$$\alpha^+ : 2^H \to 2^G, Y \mapsto f^{-1}Y.$$

Further let φ^- denote the residuated map of φ w.r.t. $(\mathbb{E}, \mathbb{D})$, that is, $(\mathbb{E}, \mathbb{D}, \varphi^-, \varphi)$ is a poset adjunction. Then, obviously, $(\mathbb{D}^{\mathrm{op}}, \mathbb{E}^{\mathrm{op}}, \varphi, \varphi^-)$ is a poset adjunction too.

For pattern structures the following operators are essential:

$$\begin{aligned} {}^{\diamond} &: 2^G \to D, X \mapsto \inf_{\mathbb{D}} \delta X \\ {}^{\blacklozenge} &: D \to 2^G, d \mapsto \{g \in G \mid d \sqsubseteq \delta g\} \\ {}^{\square} &: 2^H \to E, Z \mapsto \inf_{\mathbb{E}} \varepsilon Z \\ {}^{\blacksquare} &: E \to 2^H, e \mapsto \{h \in H \mid e \sqsubseteq \varepsilon h\} \end{aligned}$$

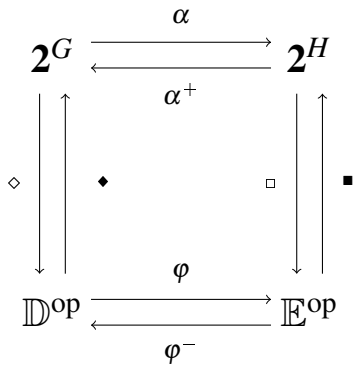
It now follows that (α, φ) forms a morphism from the poset adjunction

$$\mathcal{P} = (\mathbf{2}^G, \mathbb{D}^{\mathrm{op}}, {}^{\diamond}, {}^{\blacklozenge})$$

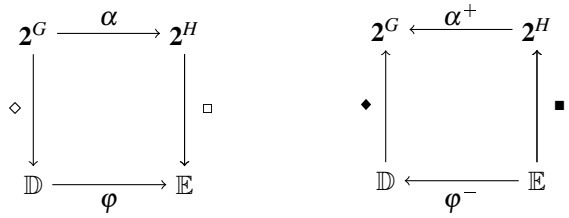
to the poset adjunction

$$\mathcal{Q} = (\mathbf{2}^H, \mathbb{E}^{\mathrm{op}}, {}^{\square}, {}^{\blacksquare}).$$

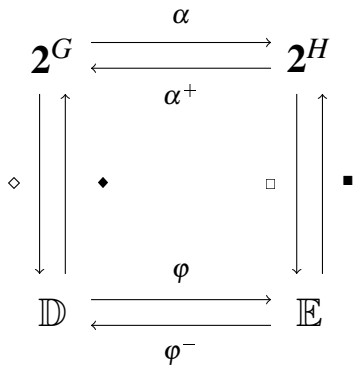
In particular, $(fX)^{\square} = \varphi(X^{\diamond})$ holds for all $X \subseteq G$.
Here we give an illustration of the constructed adjunctions:



Replacing $\mathbb{D}^{\mathrm{op}}$ by $\mathbb{D}$ and $\mathbb{E}^{\mathrm{op}}$ by $\mathbb{E}$ we receive the following commutative diagrams:



In combination we receive the following diagram of Galois connections and adjunctions between them:



For the following we recollect that the **concept lattice** of $\mathcal{G}$ is given by $\mathbb{B}\mathcal{G} := \mathbb{B}\mathcal{P}$ — similarly, $\mathbb{B}\mathcal{H} := \mathbb{B}\mathcal{Q}$.
Now we are prepared to give an application of Theorem 1 to concept lattices of pattern structures: $(\mathbb{B}\mathcal{G}, \mathbb{B}\mathcal{H}, \Phi, \Psi)$ is an adjunction for

$$\Phi : \mathrm{B}\mathcal{G} \to \mathrm{B}\mathcal{H}, (X,d) \mapsto ((\varphi d)^{\blacksquare}, \varphi d)$$

and

$$\Psi : \mathrm{B}\mathcal{H} \to \mathrm{B}\mathcal{G}, (Z, e) \mapsto (f^{-1}Z, (f^{-1}Z)^{\diamond}).$$

In case f is surjective, Φ is surjective too.
Remark: This application implies a generalization of Proposition 1 in [2], that is, if Z is an extent in $\mathcal{H}$, then $f^{-1}Z$ is an extent in $\mathcal{G}$, and if d is an intent in $\mathcal{G}$ then φd is an intent in $\mathcal{H}$.

(3) Let $\mathcal{G} = (G, \mathbb{D}, \delta)$ be a pattern structure and let κ be a kernel operator on $\mathbb{D}$. Then $\varphi : D \to \kappa D, d \mapsto \kappa d$ forms a residual map from $\mathbb{D}$ to $\kappa\mathbb{D} := \mathbb{D} \mid \kappa D$, and (id_G, φ) is a pattern morphism from $\mathcal{G}$ to $\mathcal{H} := (G, \kappa\mathbb{D}, \varphi \circ \delta)$.
Remark: In [2], φ is called an o-projection. The above clarifies the role of o-projections for pattern structures.

(4) Let $\mathcal{G} = (G, \mathbb{D}, \delta)$ be a pattern structure, and let κ be a residual kernel operator on $\mathbb{D}$. Then (id_G, κ) is a pattern morphism from $\mathcal{G}$ to $\mathcal{H} := (G, \mathbb{D}, \kappa \circ \delta)$.
Remark: In [8], κ is also referred to as a residual projection. The above clarifies the role of residual projections for pattern structures.

(5) Generalizing [2] and [8], we observe that if $\mathcal{G} = (G, \mathbb{D}, \delta)$ is a pattern structure and φ is a residual map from $\mathbb{D}$ to $\mathbb{E}$, then (id_G, φ) is a pattern morphism from $\mathcal{G}$ to $\mathcal{H} = (G, \mathbb{E}, \varphi \circ \delta)$ satisfying that

$$\Phi : \mathrm{B}\mathcal{G} \to \mathrm{B}\mathcal{H}, (X, d) \mapsto ((\varphi d)^{\blacksquare}, \varphi d)$$

is a surjective residuated map from $\mathbb{B}\mathcal{G}$ to $\mathbb{B}\mathcal{H}$.
In particular, $X^{\square} = \varphi(X^{\diamond})$ holds for all $X \subseteq G$.
Remark: This application gives a better understanding to properly generalize the concept of projections as discussed in [3] and subsequently in [2, 4–8].

References

1. T.S. Blyth, M.F.Janowitz (1972), Residuation Theory, Pergamon Press, pp. 1-382.
2. A. Buzmakov, S. O. Kuznetsov, A. Napoli (2015) , Revisiting Pattern Structure Projections. Formal Concept Analysis. Lecture Notes in Artificial Intelligence (Springer), Vol. 9113, pp 200-215.
3. B. Ganter, S. O. Kuznetsov (2001), Pattern Structures and Their Projections. Proc. 9th Int. Conf. on Conceptual Structures, ICCS01, G. Stumme and H. Delugach (Eds.). Lecture Notes in Artificial Intelligence (Springer), Vol. 2120, pp. 129-142.
4. T. B. Kaiser, S. E. Schmidt (2011), Some remarks on the relation between annotated ordered sets and pattern structures. Pattern Recognition and Machine Intelligence. Lecture Notes in Computer Science (Springer), Vol. 6744, pp 43-48.
5. M. Kaytoue, S. O. Kuznetsov, A. Napoli, S. Duplessis (2011), Mining gene expression data with pattern structures in formal concept analysis. Information Sciences (Elsevier), Vol.181, pp. 1989-2001.
6. S. O. Kuznetsov (2009), Pattern structures for analyzing complex data. In H. Sakai et al. (Eds.). Proceedings of the 12th international conference on rough sets, fuzzy sets, data mining and granular computing (RSFDGrC09). Lecture Notes in Artificial Intelligence (Springer), Vol. 5908, pp. 33-44.

7. S. O. Kuznetsov (2013), Scalable Knowledge Discovery in Complex Data with Pattern Structures. In: P. Maji, A. Ghosh, M.N. Murty, K. Ghosh, S.K. Pal, (Eds.). Proc. 5th International Conference Pattern Recognition and Machine Intelligence (PReMI2013). Lecture Notes in Computer Science (Springer), Vol. 8251, pp. 30-41.
8. L. Lumpe, S. E. Schmidt (2015), A Note on Pattern Structures and Their Projections. Formal Concept Analysis. Lecture Notes in Artificial Intelligence (Springer), Vol. 9113, pp 145-150.

NextClosures: Parallel Computation of the Canonical Base

Francesco Kriegel and Daniel Borchmann

Institute of Theoretical Computer Science, TU Dresden, Germany
{francesco.kriegel,daniel.borchmann}@tu-dresden.de

Abstract. The canonical base of a formal context plays a distinguished role in formal concept analysis. This is because it is the only minimal base so far that can be described explicitly. For the computation of this base several algorithms have been proposed. However, all those algorithms work sequentially, by computing only one pseudo-intent at a time – a fact which heavily impairs the practicability of using the canonical base in real-world applications. In this paper we shall introduce an approach that remedies this deficit by allowing the canonical base to be computed in a parallel manner. First experimental evaluations show that for sufficiently large data-sets the speedup is proportional to the number of available CPUs.

Keywords: Formal Concept Analysis, Canonical Base, Parallel Algorithms

1 Introduction

The implicational theory of a formal context is of interest in a large variety of applications. In those cases, computing the canonical base of the given context is often desirable, as it has minimal cardinality among all possible bases. On the other hand, conducting this computation often imposes a major challenge, often endangering the practicability of the underlying approach.

There are two known algorithms for computing the canonical base of a formal context [6, 12]. Both algorithms work sequentially, i.e. they compute one implication after the other. Moreover, both algorithms compute in addition to the implications of the canonical base all formal concepts of the given context. This is a disadvantage, as the number of formal concepts can be exponential in the size of the canonical base. On the other hand, the size of the canonical base can be exponential in the size of the underlying context [10]. Additionally, up to today it is not known whether the canonical base can be computed in output-polynomial time, and certain complexity results hint at a negative answer [3]. For the algorithm from [6], and indeed for any algorithm that computes the pseudo-intents in a lectic order, it has been shown that it cannot compute the canonical base with polynomial delay [2].

However, the impact of theoretical complexity results for practical application is often hard to access, and it is often worth investigating faster algorithm for theoretically intractable results. A popular approach is to explore the possibilities to parallelize known sequential algorithms. This is also true for formal concept analysis, as can be seen in the development of parallel versions for computing the concept lattice of a formal context [5, 13].

In this work we want to investigate the development of a parallel algorithm for computing the canonical base of a formal context $\mathbb{K}$. The underlying idea is actually

quite simple, and has been used by Lindig [11] to (sequentially) compute the concept lattice of a formal context: to compute the canonical base, we compute the lattice of all *intents* and *pseudo-intents* of $\mathbb{K}$. This lattice can be computed bottom up, in a level-wise order, and this computation can be done in parallel provided that the lattice has a certain "width" at a particular level. The crucial fact now is that the upper neighbours of an intent or pseudo-intent B can be easily computed by just iterating over all attributes $m \notin B$ and computing the closure of $B \cup \{ m \}$. In the approach of Lindig mentioned above this closure is just the usual double-prime operation $B \mapsto B^{II}$ of the underlying formal context $\mathbb{K}$. In our approach it is the closure operator whose closures are exactly the intents and pseudo-intents of $\mathbb{K}$. Surprisingly, despite the simpleness of our approach, we are not aware of any prior work on computing the canonical base of a formal context in a parallel manner. Furthermore, experimental results presented in this work indicate that for suitable large data-sets the computation of the canonical base can be speed up by a factor proportional to the number of available CPUs.

The paper is structured as follows. After recalling all necessary notions of formal concept analysis in Section 2, we shall describe in Section 3 our approach of computing the canonical base in parallel. Benchmarks of this algorithm are presented in Section 4, and we shall close this work with some conclusions in Section 5.

2 Preliminaries

This section gives a brief overview on the notions of formal concept analysis [7] that are used in this document. The basic structure is a *formal context* $\mathbb{K} = (G, M, I)$ consisting of a set G of *objects*, a set M of *attributes*, and an *incidence relation* $I \subseteq G \times M$. For a pair $(g, m) \in I$ we also use the infix notation $g\ I\ m$, and say that the object g *has* the attribute m. Each formal context $\mathbb{K}$ induces the *derivation operators* $\cdot^I \colon \mathfrak{P}(G) \to \mathfrak{P}(M)$ and $\cdot^I \colon \mathfrak{P}(M) \to \mathfrak{P}(G)$ that are defined as follows for object sets $A \subseteq G$ and attribute sets $B \subseteq M$:

$$A^I \coloneqq \{ m \in M \mid \forall g \in A \colon (g, m) \in I \} \text{ and } B^I \coloneqq \{ m \in M \mid \forall m \in B \colon (g, m) \in I \}.$$

In other words, A^I is the set of all attributes that all objects from A have in common, and dually B^I is the set of all objects which have all attributes from B. A *formal concept* of $\mathbb{K}$ is a pair (A, B) such that $A^I = B$ and $B = A^I$, and the set of all formal concepts of $\mathbb{K}$ is denoted by $\mathfrak{B}(\mathbb{K})$.

An *implication* over the set M is an expression of the form $X \to Y$ where $X, Y \subseteq M$. An implication $X \to Y$ over M is *valid* in $\mathbb{K}$ if $X^I \subseteq Y^I$. A set $\mathcal{L}$ of implications over M is *valid* in $\mathbb{K}$ if each implication in $\mathcal{L}$ is valid in $\mathbb{K}$. An implication $X \to Y$ *follows* from the set $\mathcal{L}$ if $X \to Y$ is valid in every context with attribute set M in which $\mathcal{L}$ is valid. Furthermore, a *model* of $X \to Y$ is a set $T \subseteq M$ such that $X \subseteq T$ implies $Y \subseteq T$. A *model* of $\mathcal{L}$ is a model of all implications in $\mathcal{L}$, and $X^{\mathcal{L}}$ is the smallest superset of X that is a model of $\mathcal{L}$. The set $X^{\mathcal{L}}$ can be computed as follows.

$$X^{\mathcal{L}} \coloneqq \bigcup\nolimits_{n \geq 1} X^{\mathcal{L}_n} \quad \text{where} \quad X^{\mathcal{L}_1} \coloneqq X \cup \bigcup \{ B \mid A \to B \in \mathcal{L} \text{ and } A \subseteq X \}$$
$$\text{and} \quad X^{\mathcal{L}_{n+1}} \coloneqq (X^{\mathcal{L}_n})^{\mathcal{L}_1} \quad \text{for all } n \in \mathbb{N}.$$

The following lemma shows some well-known equivalent statements for entailment of implications from implication sets. We will not prove them here.

Lemma 1. *Let $\mathcal{L} \cup \{ X \to Y \}$ be a set of implications over M. Then the following statements are equivalent:*

1. *$X \to Y$ follows from $\mathcal{L}$.*
2. *If $\mathbb{K}$ is a formal context with attribute set M such that $\mathcal{L}$ is valid in $\mathbb{K}$, then $X \to Y$ is also valid in $\mathbb{K}$.*
3. *If $T \subseteq M$ and T is a model of $\mathcal{L}$, then T is a model of $X \to Y$.*
4. *$Y \subseteq X^{\mathcal{L}}$.*

An attribute set $B \subseteq M$ is called *intent* of $\mathbb{K} = (G, M, I)$ if $B = B^{II}$. An attribute set $P \subseteq M$ is called *pseudo-intent* of $\mathbb{K}$ if $P \neq P^{II}$, and furthermore for each pseudo-intent $Q \subsetneq P$ the set inclusion $Q^{II} \subseteq P$ is satisfied. We denote the set of all pseudo-intents of $\mathbb{K}$ by $\mathrm{PsInt}(\mathbb{K})$. Then the *canonical implicational base* of $\mathbb{K}$ is defined as the following implication set:

$$\{ P \to P^{II} \mid P \in \mathrm{PsInt}(\mathbb{K}) \}.$$

The canonical base has the property that it is a *minimal base* of $\mathbb{K}$, i.e. it is a *base* of $\mathbb{K}$, meaning that it is a set of valid implications of $\mathbb{K}$ such that every valid implication of $\mathbb{K}$ is entailed by it. Furthermore, its cardinality is minimal among all bases of $\mathbb{K}$.

It is readily verified that a subset $X \subseteq M$ is an intent or a pseudo-intent of $\mathbb{K}$ if and only if X is a closure of the closure operator $\mathbb{K}^*$ that is defined as follows:

$$X^{\mathbb{K}^*} := \bigcup\nolimits_{n \geq 1} X^{\mathbb{K}^*_n} \quad \text{where} \quad X^{\mathbb{K}^*_1} := X \cup \bigcup \{ P^{II} \mid P \in \mathrm{PsInt}(\mathbb{K}) \text{ and } P \subsetneq X \}$$
$$\text{and} \quad X^{\mathbb{K}^*_{n+1}} := (X^{\mathbb{K}^*_n})^{\mathbb{K}^*_1} \quad \text{for all } n \in \mathbb{N}.$$

Of course, if $\mathcal{L}$ is the canonical base of $\mathbb{K}$ as described above, then both closure operators $\mathbb{K}^*$ and $\mathcal{L}^*$ coincide, where $\mathcal{L}^*$ is defined by the following equations:

$$X^{\mathcal{L}^*} := \bigcup\nolimits_{n \geq 1} X^{\mathcal{L}^*_n} \quad \text{where} \quad X^{\mathcal{L}^*_1} := X \cup \bigcup \{ B \mid A \to B \in \mathcal{L} \text{ and } A \subsetneq X \}$$
$$\text{and} \quad X^{\mathcal{L}^*_{n+1}} := (X^{\mathcal{L}^*_n})^{\mathcal{L}^*_1} \quad \text{for all } n \in \mathbb{N}.$$

3 Parallel Computation of the Canonical Base

The well-known `NextClosure` algorithm developed by Ganter [6] can be used to enumerate the implications of the canonical base. The mathematical idea behind this algorithm is to compute all intents and pseudo-intents of our formal context $\mathbb{K}$ in a certain linear order, namely the *lectic order*. As an advantage the next (pseudo-)intent is uniquely determined, but we potentially have to do backtracking in order to find it. It can be seen quite easily that those sets form a complete lattice, and the `NextClosure` algorithm uses the closure operator $\mathbb{K}^*$ of this lattice to enumerate the pseudo-intents of $\mathbb{K}$ in the lectic order. Furthermore, this algorithm is inherently sequential, i.e. it is not possible to parallelize it.

In our approach we shall not make use of the lectic order. Indeed, our algorithm will enumerate all intents and pseudo-intents of $\mathbb{K}$ in the subset-order, with no further restrictions. As a benefit we get a very easy and obvious way to parallelize this enumeration. Moreover, in multi-threaded implementations no communication between different threads is necessary. However, as it is the case with all other known algorithms

for computing the canonical base, we also have to compute all intents in addition to all pseudo-intents of the given formal context.

The main idea is very simple and works as follows. From the definition of pseudo-intents we see that in order to decide whether an attribute set $P \subseteq M$ is a pseudo-intent we only need all pseudo-intents $Q \subsetneq P$, i.e. it suffices to know all pseudo-intents with a smaller cardinality than P. This allows for the level-wise computation of the canonical base w.r.t. the subset inclusion order, i.e. we can enumerate the (pseudo-)intents w.r.t. increasing cardinality.

An algorithm that implements this idea works as follows. First we start by considering the empty set, as it is the only set with cardinality 0. Of course, the empty set must either be an intent or a pseudo-intent, and the distinction can be made by checking whether $\emptyset = \emptyset^{II}$. Then assuming inductively that all pseudo-intents with cardinality $< k$ have been determined, we can correctly decide whether a subset $P \subseteq M$ with $|P| = k$ is a pseudo-intent or not.

To compute the lattice of intents and pseudo-intents of $\mathbb{K}$ the algorithm manages a set of *candidates* that contains the (pseudo-)intents on the current level. Then, whenever a pseudo-intent P has been found, the $\subseteq$-next closure is uniquely determined by its closure P^{II} in the context $\mathbb{K}$. If an intent B has been found, then the $\subseteq$-next closures must be of the form $(B \cup \{ m \})^{\mathbb{K}^*}$, $m \notin B$. However, as we are not aware of the full implicational base of $\mathbb{K}$ yet, but only of an *approximation* $\mathcal{L}$ of it, the operators $\mathbb{K}^*$ and $\mathcal{L}^*$ do not coincide on all subsets of M. We will show that they yield the same closure for attribute sets $B \subseteq M$ with a cardinality $|B| \leq k$ if $\mathcal{L}$ contains all implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with cardinality $|P| < k$. Consequently, the $\mathcal{L}^*$-closure of a set $B \cup \{ m \}$ may not be an intent or pseudo-intent of $\mathbb{K}$. Instead they are added to the candidate list, and are processed when all pseudo-intents with smaller cardinality have been determined. We will formally prove that this technique is correct. Furthermore, the computation of all pseudo-intents and intents of cardinality k can be done in parallel, since they are independent of each other.

In summary, we shortly describe the inductive structure of the algorithm as follows: Let $\mathbb{K}$ be a finite formal context. We use four variables: k denotes the current cardinality of candidates, $\mathbf{C}$ is the set of candidates, $\mathfrak{B}$ is a set of formal concepts, and $\mathcal{L}$ is an implication set. Then the algorithm works as follows.

1. Set $k \coloneqq 0$, $\mathbf{C} \coloneqq \{ \emptyset \}$, $\mathfrak{B} \coloneqq \emptyset$, and $\mathcal{L} \coloneqq \emptyset$.
2. In parallel: For each candidate set $C \in \mathbf{C}$ with cardinality $|C| = k$ determine whether it is $\mathcal{L}^*$-closed. If not, then add its $\mathcal{L}^*$-closure to the candidate set $\mathbf{C}$, and go to Step 5.
3. If C is an intent of $\mathbb{K}$, then add the formal concept (C^I, C) to $\mathfrak{B}$. Otherwise C must be a pseudo-intent, and thus we add the formal implication $C \to C^{II}$ to the set $\mathcal{L}$, and add the formal concept (C^I, C^{II}) to the set $\mathfrak{B}$.
4. For each observed intent C^{II}, add all its upper neighbours $C^{II} \cup \{ m \}$ where $m \notin C^{II}$ to the candidate set $\mathbf{C}$.
5. Wait until all candidates of cardinality k have been processed. If $k < |M|$, then increase the candidate cardinality k by 1, and go to Step 2. Otherwise return $\mathfrak{B}$ and $\mathcal{L}$.

In order to approximate the operator $\mathcal{L}^*$ we furthermore introduce the following notion: If $\mathcal{L}$ is a set of implications, then $\mathcal{L}\restriction_k$ denotes the subset of $\mathcal{L}$ that consists of all implications whose premises have a cardinality of at most k.

Lemma 2. *Let $\mathbb{K} = (G, M, I)$ be a formal context, $\mathcal{L}$ its canonical implicational base, and $X \subseteq M$ an attribute set. Then the following statements are equivalent:*

1. *X is either an intent or a pseudo-intent of $\mathbb{K}$.*
2. *X is $\mathbb{K}^*$-closed.*
3. *X is $\mathcal{L}^*$-closed.*
4. *X is $(\mathcal{L}\restriction_{|X|-1})^*$-closed.*
5. *There is a $k \geq |X| - 1$ such that X is $(\mathcal{L}\restriction_k)^*$-closed.*
6. *For all $k \geq |X| - 1$ it holds that X is $(\mathcal{L}\restriction_k)^*$-closed.*

Proof. 1⇔2. If X is an intent or a pseudo-intent, then it is obviously $\mathbb{K}_1^*$-closed, i.e. $\mathbb{K}^*$-closed. Vice versa, if X is $\mathbb{K}^*$-closed, but no intent, then X contains the closure P^{II} of every pseudo-intent $P \subsetneq X$, and hence X must be a pseudo-intent. 2⇔3. is obvious. 3⇔4. follows directly from the fact that $P \subsetneq X$ implies $|P| < |X|$. 4⇔5. The only-if-direction is trivial. Consider $k \geq |X| - 1$ such that X is $(\mathcal{L}\restriction_k)^*$-closed. Then X contains all conclusions B where $A \to B \in \mathcal{L}$ is an implication with premise $A \subsetneq X$ such that $|A| \leq k$. Of course, $A \subsetneq X$ implies $|A| < |X|$, and thus X is $(\mathcal{L}\restriction_{|X|-1})^*$-closed as well. 4⇔6. The only-if-direction is trivial. Finally, assume that $k \geq |X| - 1$ and X is $(\mathcal{L}\restriction_{|X|-1})^*$-closed. Obviously, there are no subsets $A \subsetneq X$ with $|X| \leq |A| \leq k$, and so X must be $(\mathcal{L}\restriction_k)^*$-closed, too. □

As an immediate consequence of Lemma 2 we infer that in order to decide the $\mathbb{K}^*$-closedness of an attribute set X it suffices to know all implications in the canonical base whose premise has a lower cardinality than X.

Corollary 3. *If $\mathcal{L}$ contains all implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with $|P| < k$, and otherwise only implications with premise cardinality k, then for all attribute sets $X \subseteq M$ with $|X| \leq k$ the following statements are equivalent:*

1. *X is an intent or a pseudo-intent of $\mathbb{K}$.*
2. *X is $\mathcal{L}^*$-closed.*

This corollary allows us in a certain sense to approximate the set of all $\mathbb{K}^*$-closures w.r.t. increasing cardinality, and thus also permits the approximation of the closure operator $\mathcal{L}^*$ where $\mathcal{L}$ is the canonical base of $\mathbb{K}$. In the following Lemma 4 we will characterise the structure of the lattice of all $\mathbb{K}^*$-closures, and also give a method to compute upper neighbours. It is true that between comparable pseudo-intents there must always be an intent. In particular, the unique upper $\mathbb{K}^*$-closed neighbour of a pseudo-intent must be an intent.

Lemma 4. *Let $\mathbb{K}$ be a formal context. Then the following statements are true:*

1. *If $P \subseteq M$ is a pseudo-intent, then there is no intent or pseudo-intent strictly between P and P^{II}.*
2. *If $B \subseteq M$ is an intent, then the next intents or pseudo-intents are of the form $(B \cup \{ m \})^{\mathbb{K}^*}$ for attributes $m \notin B$.*
3. *If $X \subsetneq Y \subseteq M$ are neighbouring $\mathbb{K}^*$-closures, then $Y = (X \cup \{ m \})^{\mathbb{K}^*}$ for all attributes $m \in Y \setminus X$.*

Algorithm 1 `NextClosures` $(\mathbb{K})$

```
1  k := 0, C := { ∅ }, B := ∅, L := ∅
2  while k ≤ |M| do
3      for all C ∈ C with |C| = k do in parallel
4          C := C \ { C }
5          if C = C^{L*} then
6              if C ≠ C^{II} then
7                  L := L ∪ { C → C^{II} }
8              B := B ∪ { (C^I, C^{II}) }
9              C := C ∪ { C^{II} ∪ { m } | m ∉ C^{II} }
10         else
11             C := C ∪ { C^{L*} }
12     Wait for termination of all parallel processes.
13     k := k + 1
14 return (B, L)
```

Proof. 1. Let $P \subseteq M$ be a pseudo-intent of $\mathbb{K}$. Then for every intent B between P and P^{II}, i.e. $P \subseteq B \subseteq P^{II}$, we have $B = B^{II} = P^{II}$. Thus, there cannot be an intent strictly between P and P^{II}. Furthermore, if Q were a pseudo-intent such that $P \subsetneq Q \subseteq P^{II}$, then $P^{II} \subseteq Q$, and thus $Q = P^{II}$, a contradiction.

2. Let $B \subseteq M$ be an intent of $\mathbb{K}$, and $X \supseteq B$ an intent or pseudo-intent of $\mathbb{K}$ such that there is no other intent or pseudo-intent between them. Then $B \subseteq B \cup \{ m \} \subseteq X$ for every $m \in X \setminus B$. Thus, $B = B^{\mathbb{K}^*} \subsetneq (B \cup \{ m \})^{\mathbb{K}^*} \subseteq X^{\mathbb{K}^*} = X$. Then $(B \cup \{ m \})^{\mathbb{K}^*}$ is an intent or a pseudo-intent between B and X that strictly contains B, and hence $X = (B \cup \{ m \})^{\mathbb{K}^*}$.

3. Consider an attribute $m \in Y \setminus X$. Then $X \cup \{ m \} \subseteq Y$, and thus $X \subsetneq (X \cup \{ m \})^{\mathbb{K}^*} \subseteq Y$, as Y is already closed. Therefore, $(X \cup \{ m \})^{\mathbb{K}^*} = Y$. □

We are now ready to formulate our algorithm `NextClosures` in pseudo-code, see Algorithm 1. In the remainder of this section we shall show that this algorithm always terminates for finite formal contexts $\mathbb{K}$, and that it returns the canonical base as well as the set of all formal concepts of $\mathbb{K}$. Beforehand, let us introduce the following notation:

1. `NextClosures` is *in state* k if it has processed all candidate sets with a cardinality $\leq k$, but none of cardinality $> k$.
2. $\mathbf{C}_k$ denotes the set of candidates in state k.
3. $\mathcal{L}_k$ denotes the set of implications in state k.
4. $\mathfrak{B}_k$ denotes the set of formal concepts in state k.

Proposition 5. *Let $\mathbb{K}$ be a formal context, and assume that* `NextClosures` *has been started on $\mathbb{K}$ and is in state k. Then the following statements are true:*

1. *$\mathbf{C}_k$ contains all pseudo-intents of $\mathbb{K}$ with cardinality $k+1$, and all intents of $\mathbb{K}$ with cardinality $k+1$ whose corresponding formal concept is not already in $\mathfrak{B}_k$.*
2. *$\mathfrak{B}_k$ contains all formal concepts of $\mathbb{K}$ whose intent has cardinality $\leq k$.*
3. *$\mathcal{L}_k$ contains all implications $P \to P^{II}$ where the premise P is a pseudo-intent of $\mathbb{K}$ with cardinality $\leq k$.*
4. *Between the states k and $k+1$ an attribute set with cardinality $k+1$ is an intent or pseudo-intent of $\mathbb{K}$ if and only if it is $\mathcal{L}^*$-closed.*

Proof. We prove the statements by induction on k. The base case handles the initial state $k = -1$. Of course, $\emptyset$ is always an intent or a pseudo-intent of $\mathbb{K}$. Furthermore, it is the only attribute set of cardinality 0 and contained in the candidate set $\mathbf{C}$. As there are no sets with cardinality ≤ -1, $\mathfrak{B}_{-1}$ and $\mathcal{L}_{-1}$ trivially satisfy Statements 2 and 3. Finally, we have that $\mathcal{L}_{-1} = \emptyset$, and hence every attribute set is $\mathcal{L}^*_{-1}$-closed, in particular $\emptyset$.

We now assume that the induction hypothesis is true for k. For every implication set $\mathcal{L}$ between states k and $k+1$, i.e. $\mathcal{L}_k \subseteq \mathcal{L} \subseteq \mathcal{L}_{k+1}$, the induction hypothesis yields that $\mathcal{L}$ contains all formal implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with cardinality $\leq k$, and furthermore only implications whose premises have cardinality $k+1$ (by definition of Algorithm 1). Additionally, we know that the candidate set $\mathbf{C}$ contains all pseudo-intents P of $\mathbb{K}$ where $|P| = k+1$, and all intents B of $\mathbb{K}$ such that $|B| = k+1$ and $(B^I, B) \notin \mathfrak{B}$. Corollary 3 immediately yields the validity of Statements 2 and 3 for $k+1$, as those $\mathbb{K}^*$-closures are recognized correctly in line 5. Then $\mathcal{L}_{k+1}$ contains all implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$ with $|P| \leq k+1$, and hence each implication set $\mathcal{L}$ with $\mathcal{L}_{k+1} \subseteq \mathcal{L} \subseteq \mathcal{L}_{k+2}$ contains all those implications and furthermore only implications with a premise cardinality $k+2$. By another application of Corollary 3 we conclude that also Statement 4 is satisfied for $k+1$.

Finally, we show Statement 1 for $k+1$. Consider any $\mathbb{K}^*$-closed set X where $|X| = k+2$. Then Lemma 4 states that for all lower $\mathbb{K}^*$-neighbours Y and all $m \in X \setminus Y$ it is true that $(Y \cup \{ m \})^{\mathbb{K}^*} = X$. We proceed with a case distinction.

If there is a lower $\mathbb{K}^*$-neighbour Y which is a pseudo-intent, then Lemma 4 yields that the (unique) next $\mathbb{K}^*$-neighbour is obtained as Y^{II}, and the formal concept (Y^I, Y^{II}) is added to the set $\mathfrak{B}$ in line 8. Of course, it is true that $X = Y^{II}$.

Otherwise all lower $\mathbb{K}^*$-neighbours Y are intents, and in particular this is the case for X being a pseudo-intent by Lemma 4. Then for all these Y we have $(Y \cup \{ m \})^{\mathbb{K}^*} = X$ for all $m \in X \setminus Y$. Furthermore, all sets Z with $Y \cup \{ m \} \subsetneq Z \subsetneq X$ are not $\mathbb{K}^*$-closed. Since $X \setminus Y$ is finite, the following sequence must also be finite:

$$C_0 \coloneqq Y \cup \{ m \} \text{ and } C_{i+1} \coloneqq C_i^{\mathcal{L}^*} \text{ where } \mathcal{L}_{|C_i|-1} \subseteq \mathcal{L} \subseteq \mathcal{L}_{|C_i|}.$$

The sequence is well-defined, since implications from $\mathcal{L}_{|C_i|} \setminus \mathcal{L}_{|C_i|-1}$ have no influence on the closure of C_i. Furthermore, the sequence obviously ends with the set X, and contains no further $\mathbb{K}^*$-closed sets, and each of the sets $C_0, C_1, \ldots$ appears as a candidate during the run of the algorithm, cf. lines 9 and 11. □

From the previous result we can infer that in the last state $|M|$ the set $\mathfrak{B}$ contains all formal concepts of the input context $\mathbb{K}$, and that $\mathcal{L}$ is the canonical base of $\mathbb{K}$. Both sets are returned from Algorithm 1, and hence we can conclude that `NextClosures` is sound and complete. The following corollary summarises our results obtained so far, and also shows termination.

Corollary 6. *If the algorithm* `NextClosures` *is started on a finite formal context* $\mathbb{K}$ *as input, then it terminates, and returns both the set of all formal concepts and the canonical base of* $\mathbb{K}$ *as output.*

Proof. The second part of the statement is a direct consequence of Proposition 5. In the final state $|M|$ the set $\mathcal{L}$ contains all formal implications $P \to P^{II}$ where P is a pseudo-intent of $\mathbb{K}$. In particular, $\mathcal{L}$ is the canonical implicational base. Furthermore, the set $\mathfrak{B}$ contains all formal concepts of $\mathbb{K}$.

Finally, the computation time between states k and $k+1$ is finite, because there are only finitely many candidates of cardinality $k+1$, and the computation of closures w.r.t. the operators $\mathcal{L}^*$ and $\cdot^{II}$ can be done in finite time. As there are exactly $|M|$ states for a finite formal context, the algorithm must terminate. □

One could ask whether there are formal contexts that do not allow for a speedup in the enumeration of all intents and pseudo-intents on parallel execution. This would happen for formal contexts whose intents and pseudo-intents are linearly ordered. However, this is impossible.

Lemma 7. *Let $\mathbb{K} = (G, M, I)$ be a non-empty clarified formal context. Then the set of its intents and pseudo-intents is not linearly ordered w.r.t. subset inclusion $\subseteq$.*

Proof. Assume that $\mathbb{K} \coloneqq (G, M, I)$ with $G \coloneqq \{ g_1, \ldots, g_n \}$, $n > 0$, were a clarified formal context with intents and pseudo-intents $P_1 \subsetneq P_2 \subsetneq \ldots \subsetneq P_\ell$. In particular, then also all object intents form a chain $g_1^I \subsetneq g_2^I \subsetneq \ldots \subsetneq g_n^I$ where $n \leq \ell$. Since $\mathbb{K}$ is attribute-clarified, it follows $\left| g_{j+1}^I \setminus g_j^I \right| = 1$ for all j, and hence w.l.o.g. $M = \{ m_1, \ldots, m_n \}$, and $g_i \; I \; m_j$ iff $i \geq j$. Eventually, $\mathbb{K}$ is isomorphic to the ordinal scale $\mathbb{K}_n \coloneqq (\{ 1, \ldots, n \}, \{ 1, \ldots, n \}, \leq)$. It is easily verified that the pseudo-intents of $\mathbb{K}_n$ are either $\emptyset$, or of the form $\{ m, n \}$ where $m < n - 1$, a contradiction.

Consequently, there is no formal context with a linearly ordered set of intents and pseudo-intents. Hence, a parallel enumeration of the intents and pseudo-intents will always result in a speedup compared to a sequential enumeration.

4 Benchmarks

The purpose of this section is to show that our parallel algorithm for computing the canonical base indeed yields a speedup, both qualitatively and quantitatively, compared to the classical algorithm based on `NextClosure` [6]. To this end, we shall present the running times of our algorithm when applied to selected data-sets and with a varying number of available CPUs. We shall see that, up to a certain limit, the running time of our algorithms decreases proportional to the number of available CPUs. Furthermore, we shall also show that this speedup is not only qualitative, but indeed yields a real speedup compared to the original sequential algorithm for computing the canonical base.

The presented algorithm `NextClosures` has been integrated into *Concept Explorer FX* [8]. The implementation is a straight-forward adaption of Algorithm 1 to the programming language Java 8, and heavily uses the new Stream API and thread-safe concurrent collection classes (like `ConcurrentHashMap`). As we have described before, the processing of all candidates on the current cardinality level can be done in parallel, i.e. for each of them a separate thread is started that executes the necessary operations for lines 4 to 11 in Algorithm 1. Furthermore, as the candidates on the same level cannot affect each other, no communication between the threads is needed. More specifically, we have seen that the decision whether a candidate is an intent or a pseudo-intent is independent of all other sets with the same or a higher cardinality.

The formal contexts used for the benchmarks [1] are listed in Figure 1, and are either obtained from the *FCA Data Repository* [4] (ⓐ to ⓓ, and ⓕ to ⓟ), randomly created

[1] Readers who are interested in the test contexts should send a mail to one of the authors.

Formal Context	Objects	Attributes	Density
ⓐ car.cxt	1728	25	28 %
ⓑ mushroom.cxt	8124	119	19 %
ⓒ tic-tac-toe.cxt	958	29	34 %
ⓓ wine.cxt	178	68	20 %
ⓔ algorithms.cxt	2688	54	22 %
ⓕ o1000a10d10.cxt	1000	10	10 %
ⓖ o1000a20d10.cxt	1000	20	10 %
ⓗ o1000a36d17.cxt	1000	36	16 %
ⓘ o1000a49d14.cxt	1000	49	14 %
ⓙ o1000a50d10.cxt	1000	50	10 %
ⓚ o1000a64d12.cxt	1000	64	12 %
ⓛ o1000a81d11.cxt	1000	81	11 %
ⓜ o1000a100d10-001.cxt	1000	100	11 %
ⓝ o1000a100d10-002.cxt	1000	100	11 %
ⓞ o1000a100d10.cxt	1000	100	11 %
ⓟ o2000a81d11.cxt	2000	81	11 %
ⓠ 24.cxt	17	26	51 %
ⓡ 35.cxt	18	24	43 %
ⓢ 51.cxt	26	17	76 %
ⓣ 54.cxt	20	20	48 %
ⓤ 79.cxt	25	26	68 %

Fig. 1. Formal Contexts in Benchmarks

(ⓠ to ⓝ), or created from experimental results (ⓔ). For each of them we executed the implementation at least three times, and recorded the average computation times. The experiments were performed on the following two systems:

Taurus (1 Node of Bull HPC-Cluster, ZIH)
CPUs: 2x Intel Xeon E5-2690 with eight cores @ 2.9 GHz, RAM: 32 GB
Atlas (1 Node of Megware PC-Farm, ZIH)
CPUs: 4x AMD Opteron 6274 with sixteen cores @ 2.2 GHz, RAM: 64 GB

The benchmark results are displayed in Figure 2. The charts have both axes logarithmically scaled, to emphasise the correlation between the execution times and the number of available CPUs. We can see that the computation time is almost inverse linear proportional to the number of available CPUs, provided that the context is large enough. In this case there are enough candidates on each cardinality level for the computation to be done in parallel. However, we shall note that there are some cases where the computation times increase when utilising all available CPUs. We are currently not aware of an explanation for this exception – maybe it is due to some technical details of the platforms or the operation systems, e.g. some background tasks that are executed during the benchmark, or overhead caused by thread maintenance. Note that we did not have full system access during the experiments, but could only execute tasks by scheduling them in a batch system. Additionally, for some of the test contexts only benchmarks for a large number of CPUs could be performed, due to the time limitations on the test systems.

Furthermore, we have performed the same benchmark with small-sized contexts having at most 15 attributes. The computation times were far below one second. We have noticed that there is a certain number of available CPUs for which there is no

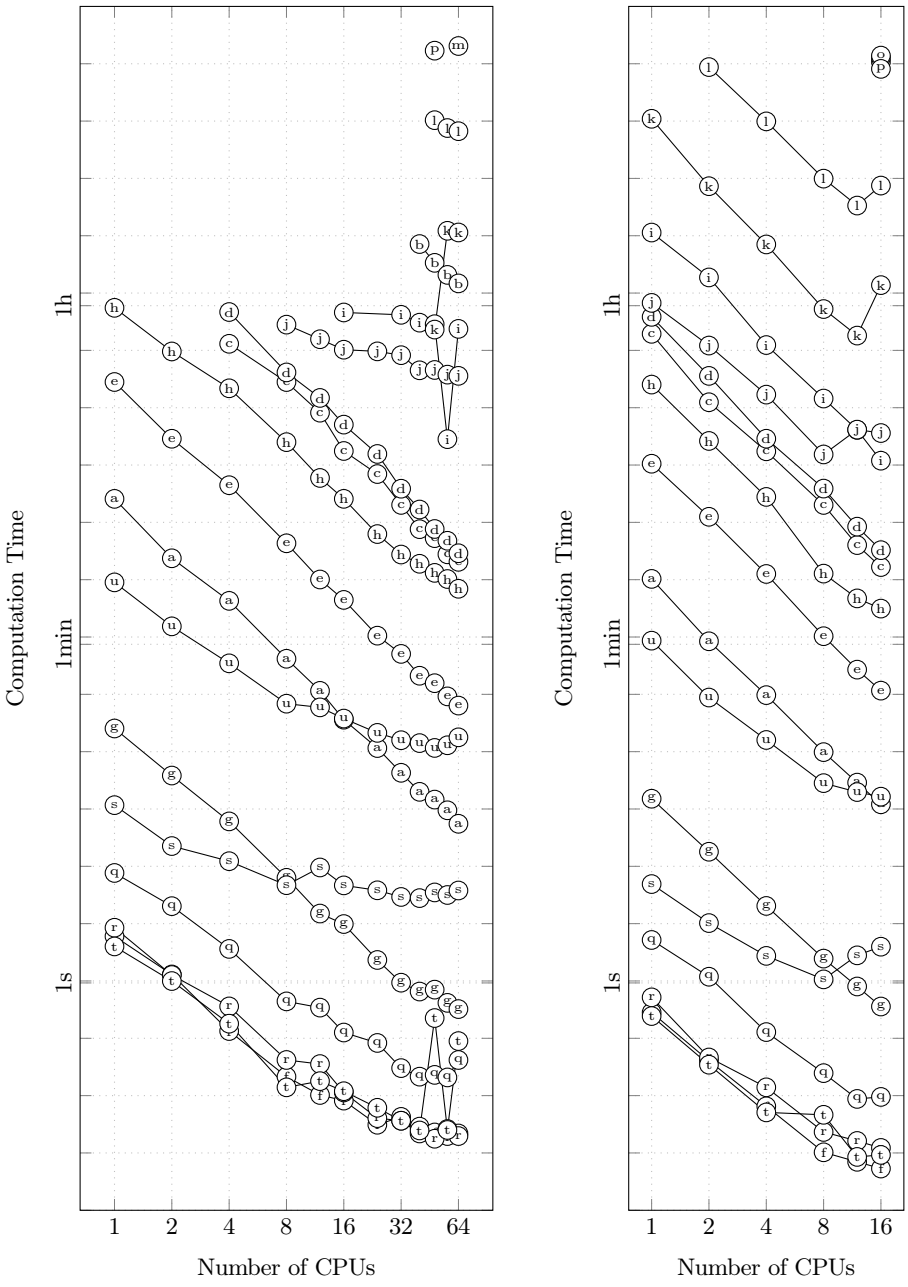


Fig. 2. Benchmark Results (left: Atlas, right: Taurus)

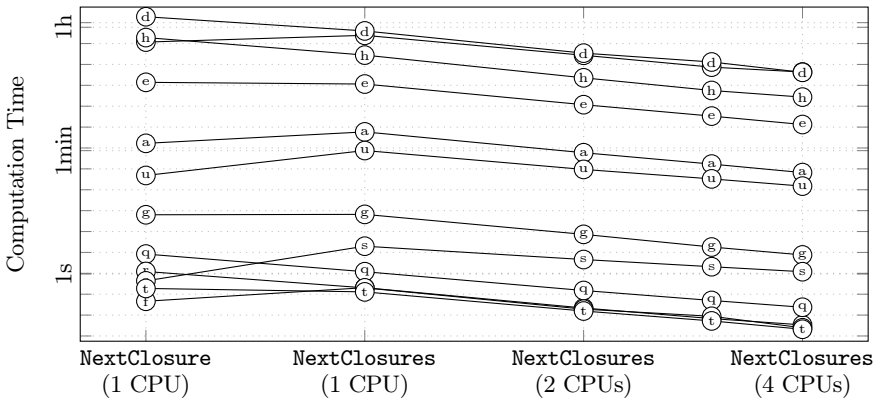


Fig. 3. Performance Comparison

further increase in speed of the algorithm. This happens when the number of candidates is smaller than the available CPUs.

Finally, we compared our two implementations of `NextClosure` and `NextClosures` when only one CPU is utilised. The comparison was performed on a notebook with Intel Core i7-3720QM CPU with four cores @ 2.6 GHz and 8 GB RAM. The results are shown in Figure 3. We conclude that our proposed algorithm is on average as fast as `NextClosure` on the test contexts. The computation time ratio is between $\frac{1}{3}$ and 3, depending on the specific context. Low or no speedups are expected for formal contexts where `NextClosure` does not have to do backtracking, and hence can find the next intent or pseudo-intent immediately.

5 Conclusion

In this paper we have introduced the parallel algorithm `NextClosures` for the computation of the canonical base. It constructs the lattice of all intents and pseudo-intents of a given formal context from bottom to top in a level-wise order w.r.t. increasing cardinality. As the elements in a certain level of this lattice can be computed independently, they can also be enumerated in parallel, thus yielding a parallel algorithm for computing the canonical base. Indeed, first benchmarks show that `NextClosures` allows for a speedup that is proportional to the number of available CPUs, up to a certain natural limit. Furthermore, we have compared its performance to the well-known algorithm `NextClosure` when utilising only one CPU. It could be observed that on average our algorithm (on one CPU) has the same performance as `NextClosure`, at least for the test contexts.

So far we have only introduced the core idea of the algorithm, but it should be clear that certain extensions are possible. For example, it is not hard to see that our parallel algorithm can be extended to also handle *background knowledge* given as a set of implications or as a *constraint closure operator* [1]. In order to yield *attribute exploration*, our algorithm can also be extended to include *expert interaction* for exploration of the canonical base of partially known contexts, much in the same way as the classical algorithm. One benefit is the possibility to have several experts answering questions in parallel. Another advantage is the constant increase in the difficulty of the questions (i.e. premise cardi-

nality), compared to the questions posed by default attribute exploration in lectic order. Those extensions have not been presented here due to a lack of space, but we shall present them in a future publication. Meanwhile, they can be found in a technical report [9].

Acknowledgements The authors thank Bernhard Ganter for helpful hints on optimal formal contexts for his `NextClosure` algorithm. Furthermore, the authors thank the anonymous reviewers for their constructive comments.

The benchmarks were performed on servers at the Institute of Theoretical Computer Science, and the Centre for Information Services and High Performance Computing (ZIH) at TU Dresden. We thank them for their generous allocations of computer time.

References

[1] Radim Belohlávek and Vilém Vychodil. "Formal Concept Analysis with Constraints by Closure Operators". In: *Conceptual Structures: Inspiration and Application, 14th International Conference on Conceptual Structures, ICCS 2006, Aalborg, Denmark, July 16-21, 2006, Proceedings*. Ed. by Henrik Schärfe, Pascal Hitzler, and Peter Øhrstrøm. Vol. 4068. Lecture Notes in Computer Science. Springer, 2006, pp. 131–143.

[2] Felix Distel. "Hardness of Enumerating Pseudo-Intents in the Lectic Order". In: *Proceedings of the 8th Interational Conference of Formal Concept Analysis*. (Agadir, Morocco). Ed. by Léonard Kwuida and Barış Sertkaya. Vol. 5986. Lecture Notes in Computer Science. Springer, 2010, pp. 124–137.

[3] Felix Distel and Barış Sertkaya. "On the Complexity of Enumerating Pseudo-Intents". In: *Discrete Applied Mathematics* 159.6 (2011), pp. 450–466.

[4] *FCA Data Repository*. URL: `http://www.fcarepository.com`.

[5] Huaiguo Fu and Engelbert Mephu Nguifo. "A Parallel Algorithm to Generate Formal Concepts for Large Data". In: *Proceedings of the Second International Conference on Formal Concept Analysis*. (Sydney, Australia). Ed. by Peter W. Eklund. Vol. 2961. Lecture Notes in Computer Science. Springer, 2004, pp. 394–401.

[6] Bernhard Ganter. "Two Basic Algorithms in Concept Analysis". In: *Proceedings of the 8th Interational Conference of Formal Concept Analysis*. (Agadir, Morocco). Ed. by Léonard Kwuida and Barış Sertkaya. Vol. 5986. Lecture Notes in Computer Science. Springer, 2010, pp. 312–340.

[7] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[8] Francesco Kriegel. *Concept Explorer FX*. Software for Formal Concept Analysis. 2010-2015. URL: `https://github.com/francesco-kriegel/conexp-fx`.

[9] Francesco Kriegel. *NextClosures – Parallel Exploration of Constrained Closure Operators*. LTCS-Report 15-01. Chair for Automata Theory, TU Dresden, 2015.

[10] Sergei O. Kuznetsov. "On the Intractability of Computing the Duquenne-Guigues Base". In: *Journal of Universal Computer Science* 10.8 (2004), pp. 927–933.

[11] Christian Lindig. "Fast Concept Analysis". In: *Working with Conceptual Structures – Contributions to ICCS 2000*. (Aachen, Germany). Ed. by Gerhard Stumme. Shaker Verlag, 2000, pp. 152–161.

[12] Sergei A. Obiedkov and Vincent Duquenne. "Attribute-Incremental Construction of the Canonical Implication Basis". In: *Annals of Mathematics and Artificial Intelligence* 49.1-4 (2007), pp. 77–99.

[13] Vilém Vychodil, Petr Krajča, and Jan Outrata. "Parallel Recursive Algorithm for FCA". In: *Proceedings of the 6th International Conference on Concept Lattices and Their Applications*. Ed. by Radim Bělohlávek and Sergej O. Kuznetsov. Palacký University, Olomouc, 2008, pp. 71–82.

Probabilistic Implicational Bases in FCA and Probabilistic Bases of GCIs in $\mathcal{EL}^\perp$

Francesco Kriegel

Institute for Theoretical Computer Science, TU Dresden, Germany
francesco.kriegel@tu-dresden.de
http://lat.inf.tu-dresden.de/~francesco

Abstract. A probabilistic formal context is a triadic context whose third dimension is a set of worlds equipped with a probability measure. After a formal definition of this notion, this document introduces probability of implications, and provides a construction for a base of implications whose probability satisfy a given lower threshold. A comparison between confidence and probability of implications is drawn, which yields the fact that both measures do not coincide, and cannot be compared. Furthermore, the results are extended towards the light-weight description logic $\mathcal{EL}^\perp$ with probabilistic interpretations, and a method for computing a base of general concept inclusions whose probability fulfill a certain lower bound is proposed.

Keywords: Formal Concept Analysis, Description Logics, Probabilistic Formal Context, Probabilistic Interpretation, Implication, General Concept Inclusion

1 Introduction

Most data-sets from real-world applications contain errors and noise. Hence, for mining them special techniques are necessary in order to circumvent the expression of the errors. This document focuses on rule mining, especially we attempt to extract rules that are approximately valid in data-sets, or families of data-sets, respectively. There are at least two measures for the approximate soundness of rules, namely *confidence* and *probability*. While confidence expresses the number of counterexamples in a single data-set, probability expresses somehow the number of data-sets in a data-set family that do not contain any counterexample. More specifically, we consider implications in the formal concept analysis setting [7], and general concept inclusions (GCIs) in the description logics setting [1] (in the light-weight description logic $\mathcal{EL}^\perp$).

Firstly, for axiomatizing rules from formal contexts possibly containing wrong incidences or having missing incidences the notion of a partial implication (also called association rule) and confidence has been defined by Luxenburger in [12]. Furthermore, Luxenburger introduced a method for the computation of a base of all partial implications holding in a formal context whose confidence is above a certain threshold. In [2] Borchmann has extended the results to the description logic $\mathcal{EL}^\perp$ by adjusting the notion of confidence to GCIs, and also gave a method for the construction of a base of confident GCIs for an interpretation.

Secondly, another perspective is a family of data-sets representing different views of the same domain, e.g., knowledge of different persons, or observations of an experiment that has been repeated several times, since some effects could not be observed

in every case. In the field of formal concept analysis, Vityaev, Demin, and Ponomaryov, have introduced in probabilistic extensions of formal contexts and their formal concepts and implications, and furthermore gave some methods for their computation, cf. [4]. In [9] the author has shown some methods for the computation of a base of GCIs in probabilistic description logics where concept and role constructors are available to express probability directly in the concept descriptions. Here, we want to use another approach, and do not allow for probabilistic constructors, but define the notion of a probability of general concept inclusions in the light-weight description logic $\mathcal{EL}^\bot$. Furthermore, we provide a method for the computation of a base of GCIs satisfying a certain lower threshold for the probability. More specifically, we use the description logic $\mathcal{EL}^\bot$ with probabilistic interpretations that have been introduced by Lutz and Schröder in [11]. Beforehand, we only consider conjunctions in the language of formal concept analysis, and define the notion of a probabilistic formal context in a more general form than in [4], and provide a technique for the computation of base of implications satisfying a given lower probability threshold.

The document is structured as follows. In Section 2 some basic notions for probabilistic extensions of formal concept analysis are defined. Then, in Section 3 a method for the computation of a base for all implications satisfying a given lower probability threshold in a probabilistic formal context is developed, and its correctness is proven. The following sections extend the results to the description logic $\mathcal{EL}^\bot$. In particular, Section 4 introduces the basic notions for $\mathcal{EL}^\bot$, and defines probabilistic interpretations. Section 5 shows a technique for the construction of a base of GCIs holding in a probabilistic interpretation and fulfilling a lower probability threshold. Furthermore, a comparison of the notions of confidence and probability is drawn at the end of Section 3.

2 Probabilistic Formal Concept Analysis

A *probability measure* $\mathbb{P}$ on a countable set W is a mapping $\mathbb{P}\colon 2^W \to [0,1]$ such that $\mathbb{P}(\emptyset) = 0$ and $\mathbb{P}(W) = 1$ hold, and $\mathbb{P}$ is *σ-additive*, i.e., for all pairwise disjoint countable families $(U_n)_{n\in\mathbb{N}}$ with $U_n \subseteq W$ it holds that $\mathbb{P}(\bigcup_{n\in\mathbb{N}} U_n) = \sum_{n\in\mathbb{N}} \mathbb{P}(U_n)$. A world $w \in W$ is *possible* if $\mathbb{P}\{\, w \,\} > 0$ holds, and *impossible* otherwise. The set of all possible worlds is denoted by W_ε, and the set of all impossible worlds is denoted by W_0. Obviously, $W_\varepsilon \uplus W_0$ is a partition of W.

Definition 1 (Probabilistic Formal Context). *A* probabilistic formal context $\mathbb{K}$ *is a tuple* $(G, M, W, I, \mathbb{P})$ *that consists of a set G of* objects*, a set M of* attributes*, a countable set W of* worlds*, an* incidence relation $I \subseteq G \times M \times W$*, and a probability measure* $\mathbb{P}$ *on W. For a triple $(g, m, w) \in I$ we say that* object g has attribute m in world w*. Furthermore, we define the* derivations in world w *as operators $\cdot^{I_w}\colon 2^G \to 2^M$ and $\cdot^{I_w}\colon 2^M \to 2^G$ where*

$$A^{I_w} \coloneqq \{\, m \in M \mid \forall g \in A\colon (g, m, w) \in I \,\}$$
$$B^{I_w} \coloneqq \{\, g \in G \mid \forall m \in B\colon (g, m, w) \in I \,\}$$

for object sets $A \subseteq G$ and attribute sets $B \subseteq M$, i.e., A^{I_w} is the set of all common attributes of all objects in A in the world w, and B^{I_w} is the set of all objects that have all attributes in B in w.

Definition 2 (Implication, Probability). *Let* $\mathbb{K} = (G, M, W, I, \mathbb{P})$ *be a probabilistic formal context. For attribute sets* $X, Y \subseteq M$ *we call* $X \to Y$ *an* implication *over* M*, and its* probability in $\mathbb{K}$ *is defined as the measure of the set of worlds it holds in, i.e.,*

$$\mathbb{P}(X \to Y) := \mathbb{P}\{\, w \in W \mid X^{I_w} \subseteq Y^{I_w} \,\}.$$

Furthermore, we define the following properties for implications $X \to Y$*:*

1. $X \to Y$ holds *in world* w *of* $\mathbb{K}$ *if* $X^{I_w} \subseteq Y^{I_w}$ *is satisfied.*
2. $X \to Y$ certainly holds *in* $\mathbb{K}$ *if it holds in all worlds of* $\mathbb{K}$.
3. $X \to Y$ almost certainly holds *in* $\mathbb{K}$ *if it holds in all possible worlds of* $\mathbb{K}$.
4. $X \to Y$ possibly holds *in* $\mathbb{K}$ *if it holds in a possible world of* $\mathbb{K}$.
5. $X \to Y$ *is* impossible *in* $\mathbb{K}$ *if it does not hold in any possible world of* $\mathbb{K}$.
6. $X \to Y$ *is* refuted *by* $\mathbb{K}$ *if does not hold in any world of* $\mathbb{K}$.

It is readily verified that $\mathbb{P}(X \to Y) = \mathbb{P}\{\, w \in W_\varepsilon \mid X^{I_w} \subseteq Y^{I_w} \,\} = \sum\{\, \mathbb{P}\{\, w \,\} \mid w \in W_\varepsilon \text{ and } X^{I_w} \subseteq Y^{I_w} \,\}$. An implication $X \to Y$ almost certainly holds if $\mathbb{P}(X \to Y) = 1$, possibly holds if $\mathbb{P}(X \to Y) > 0$, and is impossible if $\mathbb{P}(X \to Y) = 0$. If $X \to Y$ certainly holds, then it is almost certain, and if $X \to Y$ is refuted, then it is impossible.

3 Probabilistic Implicational Bases

At first we introduce the notion of a probabilistic implicational base. Then we will develop and prove a construction for such bases w.r.t. probabilistic formal contexts. If the underlying context is finite, then the base is computable. The reader should be aware of the standard notions of formal concept analysis in [7]. Recall that an implication follows from an implication set if, and only if, it can be syntactically deduced using the so-called *Armstrong rules* as follows: 1. From $X \supseteq Y$ infer $X \to Y$. 2. From $X \to Y$ and $Y \to Z$ infer $X \to Z$. 3. From $X_1 \to Y_1$ and $X_2 \to Y_2$ infer $X_1 \cup X_2 \to Y_1 \cup Y_2$.

Definition 3 (Probabilistic Implicational Base). *Let* $\mathbb{K} = (G, M, W, I, \mathbb{P})$ *be a probabilistic formal context, and* $p \in [0, 1]$ *a threshold. A* probabilistic implicational base *for* $\mathbb{K}$ *and* p *is an implication set* $\mathcal{B}$ *over* M *that satisfies the following properties:*

1. $\mathcal{B}$ *is* sound *for* $\mathbb{K}$ *and* p*, i.e.,* $\mathbb{P}(X \to Y) \geq p$ *holds for all implications* $X \to Y \in \mathcal{B}$*, and*
2. $\mathcal{B}$ *is* complete *for* $\mathbb{K}$ *and* p*, i.e., if* $\mathbb{P}(X \to Y) \geq p$*, then* $X \to Y$ *follows from* $\mathcal{B}$.

A probabilistic implicational base is irredundant *if none of its implications follows from the others, and is* minimal *if it has minimal cardinality among all bases for* $\mathbb{K}$ *and* p.

It is readily verified that the above definition is a straight-forward generalization of implicational bases as defined in [7, Definition 37], in particular formal contexts coincide with probabilistic formal contexts having only one possible world, and implications holding in the formal context coincide with implications having probability 1.

We now define a transformation from probabilistic formal contexts to formal contexts. It allows to decide whether an implication (almost) certainly holds, and furthermore it can be utilized to construct an implicational base for the (almost) certain implications.

Definition 4 (Scaling). *Let* $\mathbb{K}$ *be a probabilistic formal context. The* certain scaling *of* $\mathbb{K}$ *is the formal context* $\mathbb{K}^\times := (G \times W, M, I^\times)$ *where* $((g, w), m) \in I^\times$ *iff* $(g, m, w) \in I$*, and the* almost certain scaling *of* $\mathbb{K}$ *is the subcontext* $\mathbb{K}_\varepsilon^\times := (G \times W_\varepsilon, M, I_\varepsilon^\times)$ *of* $\mathbb{K}^\times$.

Lemma 5. *Let $\mathbb{K} = (G, M, W, I, \mathbb{P})$ be a probabilistic formal context, and let $X \to Y$ be a formal implication. Then the following statements are satisfied:*

1. *$X \to Y$ certainly holds in $\mathbb{K}$ if, and only if, $X \to Y$ holds in $\mathbb{K}^\times$.*
2. *$X \to Y$ almost certainly holds in $\mathbb{K}$ if, and only if, $X \to Y$ holds in $\mathbb{K}_\varepsilon^\times$.*

Proof. It is readily verified that the following equivalences hold:

$$\begin{aligned}\mathbb{P}(X \to Y) = 1 &\Leftrightarrow \forall w \in W \colon X^{I_w} \subseteq Y^{I_w} \\ &\Leftrightarrow X^{I^\times} = \biguplus\nolimits_{w \in W} X^{I_w} \times \{ w \} \subseteq \biguplus\nolimits_{w \in W} Y^{I_w} \times \{ w \} = Y^{I^\times} \\ &\Leftrightarrow \mathbb{K}^\times \models X \to Y.\end{aligned}$$

The second statement can be proven analogously. □

Recall the notion of a pseudo-intent [6–8]: An attribute set $P \subseteq M$ of a formal context (G, M, I) is a *pseudo-intent* if $P \neq P^{II}$, and $Q^{II} \subseteq P$ holds for all pseudo-intents $Q \subsetneq P$. Furthermore, it is well-known that the *canonical implicational base* of a formal context (G, M, I) consists of all implications $P \to P^{II}$ where P is a pseudo-intent, cf. [6–8]. Consequently, the next corollary is an immediate consequence of Lemma 5.

Corollary 6. *Let $\mathbb{K}$ be a probabilistic formal context. Then the following statements hold:*

1. *An implicational base for $\mathbb{K}^\times$ is an implicational base for the certain implications of $\mathbb{K}$, in particular this holds for the following implication set:*

$$\mathcal{B}_\mathbb{K} := \{ P \to P^{I^\times I^\times} \mid P \in \mathsf{PsInt}(\mathbb{K}^\times) \}.$$

2. *An implicational base for $\mathbb{K}_\varepsilon^\times$ w.r.t. the background knowledge $\mathcal{B}_\mathbb{K}$ is an implicational base for the almost certain implications of $\mathbb{K}$, in particular this holds for the following implication set:*

$$\mathcal{B}_{\mathbb{K},1} := \mathcal{B}_\mathbb{K} \cup \{ P \to P^{I_\varepsilon^\times I_\varepsilon^\times} \mid P \in \mathsf{PsInt}(\mathbb{K}_\varepsilon^\times, \mathcal{B}_\mathbb{K}) \}.$$

Lemma 7. *Let $\mathbb{K} = (G, M, W, I, \mathbb{P})$ be a probabilistic formal context. Then the following statements are satisfied:*

1. *$Y \subseteq X$ implies that $X \to Y$ certainly holds in $\mathbb{K}$.*
2. *$X_1 \subseteq X_2$ and $Y_1 \supseteq Y_2$ imply $\mathbb{P}(X_1 \to Y_1) \leq \mathbb{P}(X_2 \to Y_2)$.*
3. *$X_0 \subseteq X_1 \subseteq \ldots \subseteq X_n$ implies $\mathbb{P}(X_0 \to X_n) \leq \bigwedge_{i=1}^n \mathbb{P}(X_{i-1} \to X_i)$.*

Proof. 1. If $Y \subseteq X$, then $X^{I_w} \subseteq Y^{I_w}$ follows for all worlds $w \in W$.

2. Assume $X_1 \subseteq X_2$ and $Y_2 \subseteq Y_1$. Then $X_1^{I_w} \supseteq X_2^{I_w}$ and $Y_2^{I_w} \supseteq Y_1^{I_w}$ follow for all worlds $w \in W$. Consider a world $w \in W$ where $X_1^{I_w} \subseteq Y_1^{I_w}$. Of course, we may conclude that $X_2^{I_w} \subseteq Y_2^{I_w}$. As a consequence we get $\mathbb{P}(X_1 \to Y_1) \leq \mathbb{P}(X_2 \to Y_2)$.
3. We prove the third claim by induction on n. For $n = 0$ there is nothing to show, and the case $n = 1$ is trivial. Hence, consider $n = 2$ for the induction base, and let $X_0 \subseteq X_1 \subseteq X_2$. Then we have that $X_0^{I_w} \supseteq X_1^{I_w} \supseteq X_2^{I_w}$ is satisfied in all worlds $w \in W$. Now consider a world $w \in W$ where $X_0^{I_w} \subseteq X_2^{I_w}$ is true.

Of course, it then follows that $X_0^{I_w} \subseteq X_1^{I_w} \subseteq X_2^{I_w}$. Consequently, we conclude $\mathbb{P}(X_0 \to X_2) \leq \mathbb{P}(X_0 \to X_1)$ and $\mathbb{P}(X_0 \to X_2) \leq \mathbb{P}(X_1 \to X_2)$.
For the induction step let $n > 2$. The induction hypothesis yields that

$$\mathbb{P}(X_0 \to X_{n-1}) \leq \bigwedge_{i=1}^{n-1} \mathbb{P}(X_{i-1} \to X_i).$$

Of course, it also holds that $X_0 \subseteq X_{n-1} \subseteq X_n$, and it follows by induction hypothesis and the previous inequality that

$$\mathbb{P}(X_0 \to X_n) \leq \mathbb{P}(X_0 \to X_{n-1}) \wedge \mathbb{P}(X_{n-1} \to X_n) \leq \bigwedge_{i=1}^{n} \mathbb{P}(X_{i-1} \to X_i). \qquad \square$$

Lemma 8. *Let $\mathbb{K} = (G, M, W, I, \mathbb{P})$ be a probabilistic formal context. Then for all implications $X \to Y$ the following equalities are valid:*

$$\mathbb{P}(X \to Y) = \mathbb{P}(X^{I^\times I^\times} \to Y^{I^\times I^\times}) = \mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}).$$

Proof. Let $X \to Y$ be an implication. Then for all worlds $w \in W$ it holds that

$$g \in X^{I_w} \Leftrightarrow \forall m \in X\colon (g, m, w) \in I \Leftrightarrow \forall m \in X\colon ((g, w), m) \in I^\times \Leftrightarrow (g, w) \in X^{I^\times},$$

and we conclude that $X^{I_w} = \pi_1(X^{I^\times} \cap (G \times \{ w \}))$. Furthermore, we then infer $X^{I_w} = X^{I^\times I^\times I_w}$, and thus the following equations hold:

$$\begin{aligned} \mathbb{P}(X \to Y) &= \mathbb{P}\{ w \in W \mid X^{I_w} \subseteq Y^{I_w} \} \\ &= \mathbb{P}\{ w \in W \mid X^{I^\times I^\times I_w} \subseteq Y^{I^\times I^\times I_w} \} = \mathbb{P}(X^{I^\times I^\times} \to Y^{I^\times I^\times}). \end{aligned}$$

In particular, for all possible worlds $w \in W_\varepsilon$ it holds that $g \in X^{I_w} \Leftrightarrow (g, w) \in X^{I_\varepsilon^\times}$, and thus $X^{I_w} = \pi_1(X^{I_\varepsilon^\times} \cap (G \times \{ w \}))$ and $X^{I_w} = X^{I_\varepsilon^\times I_\varepsilon^\times I_w}$ are satisfied. Consequently, it may be concluded that $\mathbb{P}(X \to Y) = \mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times})$. $\square$

Lemma 9. *Let $\mathbb{K}$ be a probabilistic formal context. Then the following statements hold:*

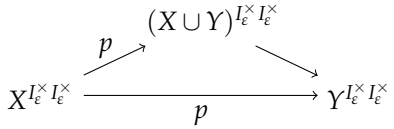
1. *If $\mathcal{B}$ is an implicational base for the certain implications of $\mathbb{K}$, then the implication $X \to Y$ follows from $\mathcal{B} \cup \{ X^{I^\times I^\times} \to Y^{I^\times I^\times} \}$.*
2. *If $\mathcal{B}$ is an implicational base for the almost certain implications of $\mathbb{K}$, then the implication $X \to Y$ follows from $\mathcal{B} \cup \{ X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times} \}$.*

Proof. Of course, the implication $X \to X^{I^\times I^\times}$ holds in $\mathbb{K}^\times$, i.e., certainly holds in $\mathbb{K}$ by Lemma 5, and hence follows from $\mathcal{B}$. Thus, the implication $X \to Y^{I^\times I^\times}$ is entailed by $\mathcal{B} \cup \{ X^{I^\times I^\times} \to Y^{I^\times I^\times} \}$, and because of $Y \subseteq Y^{I^\times I^\times}$ the claim follows.
The second statement follows analogously. $\square$

Lemma 10. *Let $\mathbb{K}$ be a probabilistic formal context. Then the following statements hold:*

1. $\mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}) = \mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times})$,
2. $(X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}$ *certainly holds in $\mathbb{K}$, and*

3. $X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}$ *is entailed by* $\{\, X^{I_\varepsilon^\times I_\varepsilon^\times} \to (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times}, (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times} \,\}$.



Proof. First note that $(X^{I_\varepsilon^\times I_\varepsilon^\times} \cup Y^{I_\varepsilon^\times I_\varepsilon^\times})^{I_\varepsilon^\times I_\varepsilon^\times} = (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times}$. As $Y^{I_\varepsilon^\times I_\varepsilon^\times}$ is a subset of $(X^{I_\varepsilon^\times I_\varepsilon^\times} \cup Y^{I_\varepsilon^\times I_\varepsilon^\times})^{I_\varepsilon^\times I_\varepsilon^\times}$, the implication $(X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}$ certainly holds in $\mathbb{K}$, cf. Statement 1 in Lemma 7.

Furthermore, we have that $X^{I_w} \subseteq Y^{I_w}$ if, and only if, $X^{I_w} \subseteq X^{I_w} \cap Y^{I_w} = (X \cup Y)^{I_w}$. Hence, the implication $X \to Y$ has the same probability as $X \to X \cup Y$. Consequently, we may conclude by means of Lemma 7 that

$$\mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}) = \mathbb{P}(X \to Y) = \mathbb{P}(X \to X \cup Y) = \mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times}).$$

Obviously, $\{\, X^{I_\varepsilon^\times I_\varepsilon^\times} \to (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times}, (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times} \,\}$ entails $X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}$. □

Lemma 11. *Let $\mathbb{K}$ be a probabilistic formal context, and X, Y be intents of $\mathbb{K}_\varepsilon^\times$ such that $X \subseteq Y$ and $\mathbb{P}(X \to Y) \geq p$. Then the following statements are true:*

1. *There is a chain of neighboring intents $X = X_0 \prec X_1 \prec X_2 \prec \ldots \prec X_n = Y$ in $\mathbb{K}_\varepsilon^\times$,*
2. *$\mathbb{P}(X_{i-1} \to X_i) \geq p$ for all $i \in \{\, 1, \ldots, n \,\}$, and*
3. *$X \to Y$ is entailed by $\{\, X_{i-1} \to X_i \mid i \in \{\, 1, \ldots, n \,\} \,\}$.*

Proof. The existence of a chain $X = X_0 \prec X_1 \prec X_2 \prec \ldots \prec X_{n-1} \prec X_n = Y$ of neighboring intents between X and Y in $\mathbb{K}_\varepsilon^\times$ follows from $X \subseteq Y$.

From Statement 3 in Lemma 7 it follows that all implications $X_{i-1} \to X_i$ have a probability of at least p in $\mathbb{K}$. It is trivial that they entail $X \to Y$. □

Theorem 12 (Probabilistic Implicational Base). *Let $\mathbb{K}$ be a probabilistic formal context, and $p \in [0, 1)$ a probability threshold. Then the following implication set is a probabilistic implicational base for $\mathbb{K}$ and p:*

$$\mathcal{B}_{\mathbb{K},p} := \mathcal{B}_{\mathbb{K},1} \cup \{\, X \to Y \mid X, Y \in \mathsf{Int}(\mathbb{K}_\varepsilon^\times) \text{ and } X \prec Y \text{ and } \mathbb{P}(X \to Y) \geq p \,\}.$$

Proof. All implications in $\mathcal{B}_{\mathbb{K},1}$ hold almost certainly in $\mathbb{K}$, and thus have probability 1. By construction, all other implications $X \to Y$ in the second subset have a probability $\geq p$. Hence, Statement 1 in Definition 3 is satisfied.

Now consider an implication $X \to Y$ over M such that $\mathbb{P}(X \to Y) \geq p$. We have to prove Statement 2 of Definition 3, i.e., that $X \to Y$ is entailed by $\mathcal{B}_{\mathbb{K},p}$.

Lemma 8 yields that both implications $X \to Y$ and $X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}$ have the same probability. Lemma 9 states that $X \to Y$ follows from $\mathcal{B}_{\mathbb{K},1} \cup \{\, X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times} \,\}$. According to Lemma 10, the implication $X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}$ follows from $\{\, X^{I_\varepsilon^\times I_\varepsilon^\times} \to (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times}, (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times} \,\}$. Furthermore, it holds that

$$\mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times}) = \mathbb{P}(X^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}) = \mathbb{P}(X \to Y) \geq p,$$

and the second implication $(X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times} \to Y^{I_\varepsilon^\times I_\varepsilon^\times}$ certainly holds, i.e., follows from $\mathcal{B}_{\mathbb{K},1}$. Finally, Lemma 11 states that there is a chain of neighboring intents of $\mathbb{K}_\varepsilon^\times$ starting at $X^{I_\varepsilon^\times I_\varepsilon^\times}$ and ending at $(X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times}$, i.e.,

$$X^{I_\varepsilon^\times I_\varepsilon^\times} = X_0^{I_\varepsilon^\times I_\varepsilon^\times} \prec X_1^{I_\varepsilon^\times I_\varepsilon^\times} \prec X_2^{I_\varepsilon^\times I_\varepsilon^\times} \prec \ldots \prec X_n^{I_\varepsilon^\times I_\varepsilon^\times} = (X \cup Y)^{I_\varepsilon^\times I_\varepsilon^\times},$$

such that all implications $X_{i-1}^{I_\varepsilon^\times I_\varepsilon^\times} \to X_i^{I_\varepsilon^\times I_\varepsilon^\times}$ have a probability $\geq p$, and are thus contained in $\mathcal{B}_{\mathbb{K},p}$. Hence, $\mathcal{B}_{\mathbb{K},p}$ entails the implication $X \to Y$. □

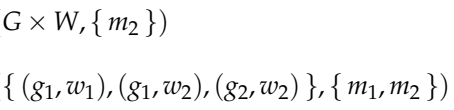
Corollary 13. *Let $\mathbb{K}$ be a probabilistic formal context. Then the following set is an implicational base for the possible implications of $\mathbb{K}$:*

$$\mathcal{B}_{\mathbb{K},\varepsilon} := \mathcal{B}_{\mathbb{K},1} \cup \{\, X \to Y \mid X, Y \in \mathsf{Int}(\mathbb{K}_\varepsilon^\times) \text{ and } X \prec Y \text{ and } \mathbb{P}(X \to Y) > 0 \,\}.$$

However, it is not possible to show irredundancy or minimality for the base of probabilistic implications given above in Theorem 12. Consider the probabilistic formal context $\mathbb{K} = (\{ g_1, g_2 \}, \{ m_1, m_2 \}, \{ w_1, w_2 \}, I, \{ \{ w_1 \} \mapsto \frac{1}{2}, \{ w_2 \} \mapsto \frac{1}{2} \})$ whose incidence relation I is defined as follows:

	m_1	m_2
g_1	×	×
g_2		×

	m_1	m_2
g_1	×	×
g_2	×	×



The only pseudo-intent of $\mathbb{K}^\times$ is $\emptyset$, and the concept lattice of $\mathbb{K}^\times$ is shown above. Hence, we have the following probabilistic implicational base for $p = \frac{1}{2}$:

$$\mathcal{B}_{\mathbb{K},\frac{1}{2}} = \{ \emptyset \to \{ m_2 \}, \{ m_2 \} \to \{ m_1, m_2 \} \}.$$

However, the set $\mathcal{B} := \{ \emptyset \to \{ m_1, m_2 \} \}$ is also a probabilistic implicational base for $\mathbb{K}$ and $\frac{1}{2}$ with less elements.

In order to compute a minimal base for the implications holding in a probabilistic formal context with a probability $\geq p$, one can for example determine the above given probabilistic base, and minimize it by means of constructing the Duquenne-Guigues base of it. This either requires the transformation of the implication set into a formal context that has this implication set as an implicational base, or directly compute all pseudo-closures of the closure operator induced by the (probabilistic) implicational base.

Recall that the confidence of an implication $X \to Y$ in a formal context (G, M, I) is defined as $\mathsf{conf}(X \to Y) := \left|(X \cup Y)^I\right| / \left|X^I\right|$, cf. [12]. In general, there is no correspondence between the probability of an implication in $\mathbb{K}$ and its confidence in $\mathbb{K}^\times$ or $\mathbb{K}_\varepsilon^\times$. To prove this we will provide two counterexamples. As first counterexample we consider the context $\mathbb{K}$ above. It is readily verified that $\mathbb{P}(\{ m_2 \} \to \{ m_1 \}) = \frac{1}{2}$ and $\mathsf{conf}(\{ m_2 \} \to \{ m_1 \}) = \frac{3}{4}$, i.e., the confidence is greater than the probability. Furthermore, consider the following modification of $\mathbb{K}$ as second counterexample:

	m_1	m_2
g_1	×	×
g_2		

	m_1	m_2
g_1		×
g_2		×

Then we have that $\mathbb{P}(\{ m_2 \} \to \{ m_1 \}) = \frac{1}{2}$ and $\mathsf{conf}(\{ m_2 \} \to \{ m_1 \}) = \frac{1}{3}$, i.e., the confidence is smaller than the probability.

4 The Description Logic $\mathcal{EL}^\bot$ and Probabilistic Interpretations

This section gives a brief overview on the light-weight description logic $\mathcal{EL}^\bot$ [1]. First, assume that (N_C, N_R) is a signature, i.e., N_C is a set of *concept names*, and N_R is a set of *role names*, respectively. Then $\mathcal{EL}^\bot$*-concept descriptions* C over (N_C, N_R) may be constructed according to the following inductive rule (where $A \in N_C$ and $r \in N_R$):

$$C ::= \bot \mid \top \mid A \mid C \sqcap C \mid \exists r.C.$$

We shall denote the set of all $\mathcal{EL}^\bot$-concept descriptions over (N_C, N_R) by $\mathcal{EL}^\bot(N_C, N_R)$. Second, the semantics of $\mathcal{EL}^\bot$-concept descriptions is defined by means of interpretations: An *interpretation* is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ that consists of a set $\Delta^\mathcal{I}$, called *domain*, and an *extension function* $\cdot^\mathcal{I} \colon N_C \cup N_R \to 2^{\Delta^\mathcal{I}} \cup 2^{\Delta^\mathcal{I} \times \Delta^\mathcal{I}}$ that maps concept names $A \in N_C$ to subsets $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and role names $r \in N_R$ to binary relations $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The extension function is extended to all $\mathcal{EL}^\bot$-concept descriptions as follows:

$$\begin{aligned} \bot^\mathcal{I} &:= \emptyset, \\ \top^\mathcal{I} &:= \Delta^\mathcal{I}, \\ (C \sqcap D)^\mathcal{I} &:= C^\mathcal{I} \cap D^\mathcal{I}, \\ (\exists r.C)^\mathcal{I} &:= \{\, d \in \Delta^\mathcal{I} \mid \exists e \in \Delta^\mathcal{I} \colon (d,e) \in r^\mathcal{I} \text{ and } e \in C^\mathcal{I} \,\}. \end{aligned}$$

A *general concept inclusion (GCI)* in $\mathcal{EL}^\bot$ is of the form $C \sqsubseteq D$ where C and D are $\mathcal{EL}^\bot$-concept descriptions. It *holds* in an interpretation $\mathcal{I}$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$ is satisfied, and we then also write $\mathcal{I} \models C \sqsubseteq D$, and say that $\mathcal{I}$ is a *model* of $C \sqsubseteq D$. Furthermore, C is *subsumed* by D if $C \sqsubseteq D$ holds in all interpretations, and we shall denote this by $C \sqsubseteq D$, too. A *TBox* is a set of GCIs, and a *model* of a TBox is a model of all its GCIs. A TBox $\mathcal{T}$ *entails* a GCI $C \sqsubseteq D$, denoted by $\mathcal{T} \models C \sqsubseteq D$, if every model of $\mathcal{T}$ is a model of $C \sqsubseteq D$.

To introduce probability into the description logic $\mathcal{EL}^\bot$, we now present the notion of a probabilistic interpretation from [11]. It is simply a family of interpretations over the same domain and the same signature, indexed by a set of worlds that is equipped with a probability measure.

Definition 14 (Probabilistic Interpretation, [11]). *Let (N_C, N_R) be a signature. A* probabilistic interpretation *$\mathcal{I}$ is a tuple $(\Delta^\mathcal{I}, (\cdot^{\mathcal{I}_w})_{w \in W}, W, \mathbb{P})$ consisting of a set $\Delta^\mathcal{I}$, called* domain*, a countable set W of* worlds*, a probability measure $\mathbb{P}$ on W, and an* extension function *$\cdot^{\mathcal{I}_w}$ for each world $w \in W$, i.e., $(\Delta^\mathcal{I}, \cdot^{\mathcal{I}_w})$ is an interpretation for each $w \in W$.*

For a general concept inclusion $C \sqsubseteq D$ its probability *in $\mathcal{I}$ is defined as follows:*

$$\mathbb{P}(C \sqsubseteq D) := \mathbb{P}\{\, w \in W \mid C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w} \,\}.$$

Furthermore, for a GCI $C \sqsubseteq D$ we define the following properties (as for probabilistic formal contexts): 1. $C \sqsubseteq D$ holds *in world w if $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$. 2. $C \sqsubseteq D$* certainly holds *in $\mathcal{I}$ if it holds in all worlds. 3. $C \sqsubseteq D$* almost certainly holds *in $\mathcal{I}$ if it holds in all possible worlds. 4. $C \sqsubseteq D$* possibly holds *in $\mathcal{I}$ if it holds in a possible world. 5. $C \sqsubseteq D$ is* impossible *in $\mathcal{I}$ if it does not hold in any possible world. 6. $C \sqsubseteq D$ is* refuted *by $\mathcal{I}$ if it does not hold in any world.*

It is readily verified that $\mathbb{P}(C \sqsubseteq D) = \mathbb{P}\{\, w \in W_\varepsilon \mid C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w} \,\} = \sum\{\, \mathbb{P}\{\, w \,\} \mid w \in W_\varepsilon \text{ and } C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w} \,\}$ for all general concept inclusions $C \sqsubseteq D$.

5 Probabilistic Bases of GCIs

In the following we construct from a probabilistic interpretation $\mathcal{I}$ a base of GCIs that entails all GCIs with a probability greater than a given threshold p w.r.t. $\mathcal{I}$.

Definition 15 (Probabilistic Base). *Let $\mathcal{I}$ be a probabilistic interpretation, and $p \in [0,1]$ a threshold. A* probabilistic base of GCIs *for $\mathcal{I}$ and p is a TBox $\mathcal{B}$ that satisfies the following conditions:*

1. *$\mathcal{B}$ is* sound *for $\mathcal{I}$ and p, i.e., $\mathbb{P}(C \sqsubseteq D) \geq p$ for all GCIs $C \sqsubseteq D \in \mathcal{B}$, and*
2. *$\mathcal{B}$ is* complete *for $\mathcal{I}$ and p, i.e., if $\mathbb{P}(C \sqsubseteq D) \geq p$, then $\mathcal{B} \models C \sqsubseteq D$.*

A probabilistic base $\mathcal{B}$ is irredundant *if none of its GCIs follows from the others, and is* minimal *if it has minimal cardinality among all probabilistic bases for $\mathcal{I}$ and p.*

For a probabilistic interpretation $\mathcal{I}$ we define its *certain scaling* as the disjoint union of all interpretations $\mathcal{I}_w$ with $w \in W$, i.e., as the interpretation $\mathcal{I}^\times := (\Delta^{\mathcal{I}} \times W, \cdot^{\mathcal{I}^\times})$ whose extension mapping is given as follows:

$$A^{\mathcal{I}^\times} := \{\, (d,w) \mid d \in A^{\mathcal{I}_w} \,\} \qquad (A \in N_C),$$
$$r^{\mathcal{I}^\times} := \{\, ((d,w),(e,w)) \mid (d,e) \in r^{\mathcal{I}_w} \,\} \qquad (r \in N_R).$$

Furthermore, the *almost certain scaling* $\mathcal{I}_\varepsilon^\times$ of $\mathcal{I}$ is the disjoint union of all interpretations $\mathcal{I}_w$ where $w \in W_\varepsilon$ is a possible world. Analogously to Lemma 5, a GCI $C \sqsubseteq D$ certainly holds in $\mathcal{I}$ iff it holds in $\mathcal{I}^\times$, and almost certainly holds in $\mathcal{I}$ iff it holds in $\mathcal{I}_\varepsilon^\times$.

In [5] the so-called *model-based most-specific concept descriptions (mmscs)* have been defined w.r.t. greatest fixpoint semantics as follows: Let $\mathcal{J}$ be an interpretation, and $X \subseteq \Delta^{\mathcal{J}}$. Then a concept description C is a *mmsc* of X in $\mathcal{J}$, if $X \subseteq C^{\mathcal{J}}$ is satisfied, and $C \sqsubseteq D$ for all concept descriptions D with $X \subseteq D^{\mathcal{J}}$. It is easy to see that all mmscs of X are unique up to equivalence, and hence we denote *the* mmsc of X in $\mathcal{J}$ by $X^{\mathcal{J}}$. Please note that there is also a role-depth bounded variant w.r.t. descriptive semantics given in [3].

Lemma 16. *Let $\mathcal{I}$ be a probabilistic interpretation. Then the following statements hold:*

1. *$C^{\mathcal{I}_w} \times \{\, w \,\} = C^{\mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\})$ for all concept descriptions C and worlds $w \in W$.*
2. *$C^{\mathcal{I}_w} \times \{\, w \,\} = C^{\mathcal{I}_\varepsilon^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\})$ for all concept descriptions C and possible worlds $w \in W_\varepsilon$.*
3. *$\mathbb{P}(C \sqsubseteq D) = \mathbb{P}(C^{\mathcal{I}^\times \mathcal{I}^\times} \sqsubseteq D^{\mathcal{I}^\times \mathcal{I}^\times}) = \mathbb{P}(C^{\mathcal{I}_\varepsilon^\times \mathcal{I}_\varepsilon^\times} \sqsubseteq D^{\mathcal{I}_\varepsilon^\times \mathcal{I}_\varepsilon^\times})$ for all GCIs $C \sqsubseteq D$.*

Proof. 1. We prove the claim by structural induction on C. By definition, the statement holds for $\bot$, $\top$, and all concept names $A \in N_C$. Consider a conjunction $C \sqcap D$, then

$$\begin{aligned}(C \sqcap D)^{\mathcal{I}_w} \times \{\, w \,\} &= (C^{\mathcal{I}_w} \cap D^{\mathcal{I}_w}) \times \{\, w \,\} \\ &= C^{\mathcal{I}_w} \times \{\, w \,\} \cap D^{\mathcal{I}_w} \times \{\, w \,\} \\ &\overset{\text{I.H.}}{=} C^{\mathcal{I}^\times} \cap D^{\mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\}) \\ &= (C \sqcap D)^{\mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\}).\end{aligned}$$

For an existential restriction $\exists\, r.\, C$ the following equalities hold:

$$\begin{aligned}
&(\exists\, r.\, C)^{\mathcal{I}_w} \times \{\, w \,\} \\
&= \{\, d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} \colon (d,e) \in r^{\mathcal{I}_w} \text{ and } e \in C^{\mathcal{I}_w} \,\} \times \{\, w \,\} \\
&= \{\, (d,w) \mid \exists (e,w) \colon ((d,w),(e,w)) \in r^{\mathcal{I}^\times} \text{ and } (e,w) \in C^{\mathcal{I}_w} \times \{\, w \,\} \,\} \\
&\overset{\text{I.H.}}{=} \{\, (d,w) \mid \exists (e,w) \colon ((d,w),(e,w)) \in r^{\mathcal{I}^\times} \text{ and } (e,w) \in C^{\mathcal{I}^\times} \,\} \\
&= (\exists\, r.\, C)^{\mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\}).
\end{aligned}$$

2. analogously.
3. Using the first statement we may conclude that the following equalities hold:

$$\begin{aligned}
&\mathbb{P}(C \sqsubseteq D) \\
&= \mathbb{P}\{\, w \in W \mid C^{\mathcal{I}_w} \times \{\, w \,\} \subseteq D^{\mathcal{I}_w} \times \{\, w \,\} \,\} \\
&= \mathbb{P}\{\, w \in W \mid C^{\mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\}) \subseteq D^{\mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\}) \,\} \\
&= \mathbb{P}\{\, w \in W \mid C^{\mathcal{I}^\times \mathcal{I}^\times \mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\}) \subseteq D^{\mathcal{I}^\times \mathcal{I}^\times \mathcal{I}^\times} \cap (\Delta^{\mathcal{I}} \times \{\, w \,\}) \,\} \\
&= \mathbb{P}\{\, w \in W \mid C^{\mathcal{I}^\times \mathcal{I}^\times \mathcal{I}_w} \times \{\, w \,\} \subseteq D^{\mathcal{I}^\times \mathcal{I}^\times \mathcal{I}_w} \times \{\, w \,\} \,\} \\
&= \mathbb{P}(C^{\mathcal{I}^\times \mathcal{I}^\times} \sqsubseteq D^{\mathcal{I}^\times \mathcal{I}^\times}).
\end{aligned}$$

The second equality follows analogously. □

For a probabilistic interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, W, \mathbb{P})$ and a set M of $\mathcal{EL}^\perp$-concept descriptions we define their *induced context* as the probabilistic formal context $\mathbb{K}_{\mathcal{I},M} \coloneqq (\Delta^{\mathcal{I}}, M, W, I, \mathbb{P})$ where $(d, C, w) \in I$ iff $d \in C^{\mathcal{I}_w}$.

Lemma 17. *Let $\mathcal{I}$ be a probabilistic interpretation, M a set of $\mathcal{EL}^\perp$-concept descriptions, and $X, Y \subseteq M$. Then the probability of the implication $X \to Y$ in the induced context $\mathbb{K}_{\mathcal{I},M}$ equals the probability of the GCI $\bigsqcap X \sqsubseteq \bigsqcap Y$ in $\mathcal{I}$, i.e., it holds that $\mathbb{P}(X \to Y) = \mathbb{P}(\bigsqcap X \sqsubseteq \bigsqcap Y)$.*

Proof. The following equivalences are satisfied for all $Z \subseteq M$ and worlds $w \in W$:

$$d \in Z^{I_w} \Leftrightarrow \forall C \in Z \colon (d, C, w) \in I \Leftrightarrow \forall C \in Z \colon d \in C^{\mathcal{I}_w} \Leftrightarrow d \in (\textstyle\bigsqcap Z)^{\mathcal{I}_w}.$$

Now consider two subsets $X, Y \subseteq M$, then it holds that

$$\begin{aligned}
\mathbb{P}(X \to Y) &= \mathbb{P}\{\, w \in W \mid X^{I_w} \subseteq Y^{I_w} \,\} \\
&= \mathbb{P}\{\, w \in W \mid (\textstyle\bigsqcap X)^{\mathcal{I}_w} \subseteq (\bigsqcap Y)^{\mathcal{I}_w} \,\} = \mathbb{P}(\textstyle\bigsqcap X \sqsubseteq \bigsqcap Y).
\end{aligned}$$

□

Analogously to [5], the context $\mathbb{K}_{\mathcal{I}}$ is defined as $\mathbb{K}_{\mathcal{I},M_{\mathcal{I}}}$ with the following attributes:

$$M_{\mathcal{I}} \coloneqq \{\, \bot \,\} \cup N_C \cup \{\, \exists\, r.\, X^{\mathcal{I}_\varepsilon^\times} \mid \emptyset \neq X \subseteq \Delta^{\mathcal{I}} \times W_\varepsilon \,\}.$$

For an implication set $\mathcal{B}$ over a set M of $\mathcal{EL}^\perp$-concept descriptions we define its *induced TBox* by $\bigsqcap \mathcal{B} \coloneqq \{\, \bigsqcap X \sqsubseteq \bigsqcap Y \mid X \to Y \in \mathcal{B} \,\}$.

Corollary 18. *If $\mathcal{B}$ contains an almost certain implicational base for $\mathbb{K}_\mathcal{I}$, then $\bigsqcap\mathcal{B}$ is complete for the almost certain GCIs of $\mathcal{I}$.*

Proof. We know that a GCI almost certainly holds in $\mathcal{I}$ if, and only if, it holds in $\mathcal{I}_\varepsilon^\times$. Let $\mathcal{B}' \subseteq \mathcal{B}$ be an almost certain implicational base for $\mathbb{K}_\mathcal{I}$, i.e., an implicational base for $(\mathbb{K}_\mathcal{I})_\varepsilon^\times = \mathbb{K}_{\mathcal{I}_\varepsilon^\times}$. Then according to Distel in [5, Theorem 5.12] it follows that the TBox $\bigsqcap\mathcal{B}'$ is a base of GCIs for $\mathcal{I}_\varepsilon^\times$, i.e., a base for the almost certain GCIs of $\mathcal{I}$. Consequently, $\bigsqcap\mathcal{B}$ is complete for the almost certain GCIs of $\mathcal{I}$. □

Theorem 19. *Let $\mathcal{I}$ be a probabilistic interpretation, and $p \in [0,1]$ a threshold. If $\mathcal{B}$ is a probabilistic implicational base for $\mathbb{K}_\mathcal{I}$ and p that contains an almost certain implicational base for $\mathbb{K}_\mathcal{I}$, then $\bigsqcap\mathcal{B}$ is a probabilistic base of GCIs for $\mathcal{I}$ and p.*

Proof. Consider a GCI $\bigsqcap X \sqsubseteq \bigsqcap Y \in \bigsqcap\mathcal{B}$. Then Lemma 17 yields that the implication $X \to Y$ and the GCI $\bigsqcap X \sqsubseteq \bigsqcap Y$ have the same probability. Since $\mathcal{B}$ is a probabilistic implicational base for $\mathbb{K}_\mathcal{I}$ and p, we conclude that $\mathbb{P}(\bigsqcap X \sqsubseteq \bigsqcap Y) \geq p$ is satisfied.

Assume that $C \sqsubseteq D$ is an arbitrary GCI with probability $\geq p$. We have to show that $\bigsqcap\mathcal{B}$ entails $C \sqsubseteq D$. Let $\mathcal{J}$ be an arbitrary model of $\bigsqcap\mathcal{B}$. Consider an implication $X \to Y \in \mathcal{B}$, then $\bigsqcap X \sqsubseteq \bigsqcap Y \in \bigsqcap\mathcal{B}$ holds, and hence it follows that $(\bigsqcap X)^\mathcal{J} \subseteq (\bigsqcap Y)^\mathcal{J}$. Consequently, the implication $X \to Y$ holds in the induced context $\mathbb{K}_{\mathcal{J},M_\mathcal{I}}$. (We here mean the non-probabilistic formal context that is induced by a non-probabilistic interpretation, cf. [2, 3, 5].)

Furthermore, since all model-based most-specific concept descriptions of $\mathcal{I}_\varepsilon^\times$ are expressible in terms of $M_\mathcal{I}$, we have that $E \equiv \bigsqcap \pi_{M_\mathcal{I}}(E)$ holds for all mmscs E of $\mathcal{I}_\varepsilon^\times$, cf. [2, 3, 5]. Hence, we may conclude that

$$\begin{aligned}\mathbb{P}(C \sqsubseteq D) &= \mathbb{P}(C^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times} \sqsubseteq D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}) \\ &= \mathbb{P}(\bigsqcap \pi_{M_\mathcal{I}}(C^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}) \sqsubseteq \bigsqcap \pi_{M_\mathcal{I}}(D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times})) \\ &= \mathbb{P}(\pi_{M_\mathcal{I}}(C^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}) \to \pi_{M_\mathcal{I}}(D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times})).\end{aligned}$$

Consequently, $\mathcal{B}$ entails the implication $\pi_{M_\mathcal{I}}(C^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}) \to \pi_{M_\mathcal{I}}(D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times})$, hence it holds in $\mathbb{K}_{\mathcal{J},M_\mathcal{I}}$, and furthermore the GCI $C^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times} \sqsubseteq D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}$ holds in $\mathcal{J}$. As $\mathcal{J}$ is an arbitrary interpretation, $\bigsqcap\mathcal{B}$ entails $C^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times} \sqsubseteq D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}$.

Corollary 18 yields that $\bigsqcap\mathcal{B}$ is complete for the almost certain GCIs of $\mathcal{I}$. In particular, the GCI $C \sqsubseteq C^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}$ almost certainly holds in $\mathcal{I}$, and hence follows from $\bigsqcap\mathcal{B}$. We conclude that $\bigsqcap\mathcal{B} \models C \sqsubseteq D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times}$. Of course, the GCI $D^{\mathcal{I}_\varepsilon^\times\mathcal{I}_\varepsilon^\times} \sqsubseteq D$ holds in all interpretations. Finally, we conclude that $\bigsqcap\mathcal{B}$ entails $C \sqsubseteq D$. □

Corollary 20. *Let $\mathcal{I}$ be a probabilistic interpretation, and $p \in [0,1]$ a threshold. Then $\bigsqcap\mathcal{B}_{\mathbb{K}_\mathcal{I},p}$ is a probabilistic base of GCIs for $\mathcal{I}$ and p where $\mathcal{B}_{\mathbb{K}_\mathcal{I},p}$ is defined as in Theorem 12.*

6 Conclusion

We have introduced the notion of a probabilistic formal context as a triadic context whose third dimension is a set of worlds equipped with a probability measure. Then

the probability of implications in such probabilistic formal contexts was defined, and a construction of a base of implications whose probability exceeds a given threshold has been proposed, and its correctness has been verified. Furthermore, the results have been applied to the light-weight description logic $\mathcal{EL}^\bot$ with probabilistic interpretations, and so we formulated a method for the computation of a base of general concept inclusions whose probability satisfies a given lower threshold.

For finite input data-sets all of the provided constructions are computable. In particular, [3, 5] provide methods for the computation of model-based most-specific concept descriptions, and the algorithms in [6, 10] can be utilized to compute concept lattices and canonical implicational bases (or bases of GCIs, respectively).

The author thanks Sebastian Rudolph for proof reading and a fruitful discussion, and the anonymous reviewers for their constructive comments.

References

[1] Franz Baader et al., eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2003.

[2] Daniel Borchmann. "Learning Terminological Knowledge with High Confidence from Erroneous Data". PhD thesis. TU Dresden, Germany, 2014.

[3] Daniel Borchmann, Felix Distel, and Francesco Kriegel. *Axiomatization of General Concept Inclusions from Finite Interpretations*. LTCS-Report 15-13. Chair for Automata Theory, Institute for Theoretical Computer Science, TU Dresden, Germany, 2015.

[4] Alexander V. Demin, Denis K. Ponomaryov, and Evgenii Vityaev. "Probabilistic Concepts in Formal Contexts". In: *Perspectives of Systems Informatics - 8th International Andrei Ershov Memorial Conference, PSI 2011, Novosibirsk, Russia, June 27-July 1, 2011, Revised Selected Papers*. Ed. by Edmund M. Clarke, Irina Virbitskaite, and Andrei Voronkov. Vol. 7162. Lecture Notes in Computer Science. Springer, 2011, pp. 394–410.

[5] Felix Distel. "Learning Description Logic Knowledge Bases from Data using Methods from Formal Concept Analysis". PhD thesis. TU Dresden, Germany, 2011.

[6] Bernhard Ganter. "Two Basic Algorithms in Concept Analysis". In: *Formal Concept Analysis, 8th International Conference, ICFCA 2010, Agadir, Morocco, March 15-18, 2010. Proceedings*. Ed. by Léonard Kwuida and Baris Sertkaya. Vol. 5986. Lecture Notes in Computer Science. Springer, 2010, pp. 312–340.

[7] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999.

[8] Jean-Luc Guigues and Vincent Duquenne. "Famille minimale d'implications informatives résultant d'un tableau de données binaires". In: *Mathématiques et Sciences Humaines* 95 (1986), pp. 5–18.

[9] Francesco Kriegel. "Axiomatization of General Concept Inclusions in Probabilistic Description Logics". In: *Proceedings of the 38th German Conference on Artificial Intelligence, KI 2015, Dresden, Germany, September 21-25, 2015*. Vol. 9324. Lecture Notes in Artificial Intelligence. Springer Verlag, 2015.

[10] Francesco Kriegel. *NextClosures – Parallel Exploration of Constrained Closure Operators*. LTCS-Report 15-01. Chair for Automata Theory, Institute for Theoretical Computer Science, TU Dresden, Germany, 2015.

[11] Carsten Lutz and Lutz Schröder. "Probabilistic Description Logics for Subjective Uncertainty". In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. Ed. by Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski. AAAI Press, 2010.

[12] Michael Luxenburger. "Implikationen, Abhängigkeiten und Galois Abbildungen". PhD thesis. TH Darmstadt, Germany, 1993.

Category of isotone bonds between L-fuzzy contexts over different structures of truth degrees

Jan Konecny[1] and Ondrej Krídlo[2]

[1] Data Analysis and Modeling Lab
Dept. Computer Science, Palacky University, Olomouc
17. listopadu 12, CZ-77146 Olomouc, Czech Republic

[2] Institute of Computer Science, Faculty of Science
Pavol Jozef Šafárik University in Košice
Jesenná 5, 040 01 Košice, Slovakia.
jan.konecny@upol.cz ondrej.kridlo@upjs.sk

Abstract. We describe properties of compositions of isotone bonds between **L**-fuzzy contexts over different complete residuated lattices and we show that **L**-fuzzy contexts as objects and isotone bonds as arrows form a category.

1 Introduction

In Formal Concept Analysis, bonds represent relationships between formal contexts. One of the motivations for introducing this notion is to provide a tool for studying mappings between formal contexts, corresponding to the behavior of Galois connections between their corresponding concept lattices. The notions of bonds, scale measures and informorphisms were studied by [14] aiming at a thorough study of the theory of morphisms in FCA.

In our previous works, we studied generalizations of bonds into an L-fuzzy setting in [12, 11]. In [13] we also provided a study of bonds between formal fuzzy contexts over different structures of truth degrees. The bonds were based on mappings between complete residuated lattices, called residuation-preserving Galois connections. These mappings were too strict and in [9] we proposed to replace them by residuation-preserving (l, k)-connections or residuation-preserving dual (l, k)-connections between complete residuated lattices.

In the present paper we continue our study [12] of properties of bonds between formal contexts over different structures of truth degrees; this time we concern with bonds mimicking isotone Galois connections between concept lattices formed by isotone concept-forming operators. Particularly, we describe the category of formal fuzzy contexts and isotone bonds between them. The paper also extends [13, 9] as we consider a setting with fuzzy formal contexts over different complete residuated lattices.

The structure of the paper is as follows. First, in Section 2 we recall basic notions required in the rest of the paper. Section 3.1 considers weak homogeneous **L**-bonds w.r.t. isotone concept-forming operators and their compositions. Section 3.2 then generalizes the results to the setting of formal fuzzy contexts over different structure of truth degrees. Finally, we summarize our results and outline our future research in this area in Section 4.

2 Preliminaries

2.1 Residuated lattices, fuzzy sets, and fuzzy relations

We use complete residuated lattices as basic structures of truth degrees. A complete residuated lattice is a structure $\mathbf{L} = \langle L, \wedge, \vee, \otimes, \to, 0, 1 \rangle$ such that

(i) $\langle L, \wedge, \vee, 0, 1 \rangle$ is a complete lattice, i.e. a partially ordered set in which arbitrary infima and suprema exist;

(ii) $\langle L, \otimes, 1 \rangle$ is a commutative monoid, i.e. $\otimes$ is a binary operation which is commutative, associative, and $a \otimes 1 = a$ for each $a \in L$;

(iii) $\otimes$ and $\to$ satisfy adjointness, i.e. $a \otimes b \leqslant c$ iff $a \leqslant b \to c$.

0 and 1 denote the least and greatest elements. The partial order of **L** is denoted by $\leqslant$. Throughout this work, **L** denotes an arbitrary complete residuated lattice.

Elements a of L are called truth degrees. Operations $\otimes$ (multiplication) and $\to$ (residuum) play the role of (truth functions of) "fuzzy conjunction" and "fuzzy implication".

An **L**-set (or **L**-fuzzy set) A in a universe set X is a mapping assigning to each $x \in X$ some truth degree $A(x) \in L$. The set of all **L**-sets in a universe X is denoted L^X.

The operations with **L**-sets are defined componentwise. For instance, the intersection of **L**-sets $A, B \in L^X$ is an **L**-set $A \cap B$ in X such that $(A \cap B)(x) = A(x) \wedge B(x)$ for each $x \in X$, etc.

An **L**-set $A \in L^X$ is called crisp if $A(x) \in \{0, 1\}$ for each $x \in X$. Crisp **L**-sets can be identified with ordinary sets. For a crisp A, we also write $x \in A$ for $A(x) = 1$ and $x \notin A$ for $A(x) = 0$. An **L**-set $A \in L^X$ is called empty (denoted by $\varnothing$) if $A(x) = 0$ for each $x \in X$. For $a \in L$ and $A \in L^X$, the a-multiplication $a \otimes A$ and a-shift $a \to A$ are **L**-sets defined by

$$(a \otimes A)(x) = a \otimes A(x),$$
$$(a \to A)(x) = a \to A(x).$$

Binary **L**-relations (binary **L**-fuzzy relations) between X and Y can be thought of as **L**-sets in the universe $X \times Y$. That is, a binary **L**-relation $I \in L^{X \times Y}$ between a set X and a set Y is a mapping assigning to each $x \in X$ and each $y \in Y$ a truth degree $I(x, y) \in L$ (a degree to which x and y are related by I).

For an **L**-relation $I \in L^{X \times Y}$ we define its transpose as the **L**-relation $I^{\mathrm{T}} \in L^{Y \times X}$ given by $I^{\mathrm{T}}(y, x) = I(x, y)$ for each $x \in X, y \in Y$.

Various composition operators for binary **L**-relations were extensively studied by [6]; we will use the following composition operators, defined for relations $A \in L^{X\times F}$ and $B \in L^{F\times Y}$:

$$(A \circ B)(x,y) = \bigvee_{f\in F} A(x,f) \otimes B(f,y), \tag{1}$$

$$(A \triangleright B)(x,y) = \bigwedge_{f\in F} B(f,y) \to A(x,f). \tag{2}$$

Note also that for $L = \{0,1\}$, $A \circ B$ coincides with the well-known composition of binary relations.

We will occasionally use some of the following properties concerning the associativity of several composition operators, see [2].

Theorem 1. *The operator $\circ$ from above has the following properties concerning composition.*

- *Associativity:*

$$R \circ (S \circ T) = (R \circ S) \circ T. \tag{3}$$

- *Distributivity:*

$$(\bigcup_i R_i) \circ S = \bigcup_i (R_i \circ S), \quad \textit{and} \quad R \circ (\bigcup_i S_i) = \bigcup_i (R \circ S_i). \tag{4}$$

2.2 Formal fuzzy concept analysis

An **L**-context is a triplet $\langle X, Y, I\rangle$ where X and Y are (ordinary nonempty) sets and $I \in L^{X\times Y}$ is an **L**-relation between X and Y. Elements of X are called objects, elements of Y are called attributes, I is called an incidence relation. $I(x,y) = a$ is read: "The object x has the attribute y to degree a."

Consider the following pair $\langle \cap, \cup\rangle$ of operators $\cap : L^X \to L^Y$ and $\cup : L^Y \to L^X$ induced by an **L**-context $\langle X, Y, I\rangle$:

$$A^{\cap}(y) = \bigvee_{x\in X} A(x) \otimes I(x,y), \quad B^{\cup}(x) = \bigwedge_{y\in Y} I(x,y) \to B(y). \tag{5}$$

for all $A \in L^X$ and $B \in \mathbf{L}^Y$. When we consider concept-forming operators induced by multiple **L**-relations, we write the inducing **L**-relation as the subscript of the symbols of the operators. For example, the pair of concept-forming operators induced by **L**-relation I are written as $\langle \cap_I, \cup_I\rangle$.

Remark 1. Notice that the pair of concept-forming operators can be interpreted as instances of the composition operators between relations. Applying the isomorphisms $\mathbf{L}^{1\times X} \cong \mathbf{L}^X$ and $\mathbf{L}^{Y\times 1} \cong \mathbf{L}^Y$ whenever necessary, one could write them, alternatively, as

$$A^{\cap} = A \circ I \quad \text{and} \quad B^{\cup} = I \triangleleft B \quad (= B \triangleright I^{\mathrm{T}}).$$

Furthermore, denote the set of fixed points of $\langle {}^{\uparrow}, {}^{\downarrow} \rangle$ by $\mathcal{B}^{\uparrow\downarrow}(X, Y, I)$, i.e.

$$\mathcal{B}^{\uparrow\downarrow}(X, Y, I) = \{\langle A, B \rangle \in L^X \times L^Y \mid A^{\uparrow} = B,\ B^{\downarrow} = A\}. \tag{6}$$

The set of fixed points endowed with $\leqslant$, defined by

$$\langle A_1, B_1 \rangle \leqslant \langle A_2, B_2 \rangle \quad \text{if } A_1 \subseteq A_2 \text{ (equivalently } B_2 \subseteq B_1)$$

is a complete lattice [5], called an *attribute-oriented* **L**-*concept lattice* associated with I, and its elements are called (*attribute-oriented*) *formal* **L**-*concepts* (or just **L**-concepts). For thorough studies of attribute-oriented concept lattices, see [5, 7, 15]. In a formal concept $\langle A, B \rangle$, the A is called an *extent*, and B is called an *intent*. The set of all extents and the set of all intents are denoted by $\mathrm{Ext}^{\uparrow\downarrow}$ and $\mathrm{Int}^{\uparrow\downarrow}$, respectively. That is,

$$\begin{aligned} \mathrm{Ext}^{\uparrow\downarrow}(X, Y, I) &= \{A \in L^X \mid \langle A, B \rangle \in \mathcal{B}^{\uparrow\downarrow}(X, Y, I) \text{ for some } B\}, \\ \mathrm{Int}^{\uparrow\downarrow}(X, Y, I) &= \{B \in L^Y \mid \langle A, B \rangle \in \mathcal{B}^{\uparrow\downarrow}(X, Y, I) \text{ for some } A\}. \end{aligned} \tag{7}$$

Equivalently, we can characterize $\mathrm{Ext}^{\uparrow\downarrow}(X, Y, I)$ and $\mathrm{Int}^{\uparrow\downarrow}(X, Y, I)$ as follows

$$\begin{aligned} \mathrm{Ext}^{\uparrow\downarrow}(X, Y, I) &= \{B^{\downarrow} \mid B \in L^Y\}, \\ \mathrm{Int}^{\uparrow\downarrow}(X, Y, I) &= \{A^{\uparrow} \mid A \in L^X\}. \end{aligned} \tag{8}$$

We will need the following lemma from [4].

Lemma 1. *Consider* **L**-*contexts* $\langle X, Y, I \rangle$, $\langle X, F, A \rangle$, *and* $\langle F, Y, B \rangle$.

(a) $\mathrm{Int}^{\uparrow\downarrow}(X, Y, I) \subseteq \mathrm{Int}^{\uparrow\downarrow}(F, Y, B)$ *if and only if there exists* $A' \in L^{X \times F}$ *such that* $I = A' \circ B$,
(b) $\mathrm{Ext}^{\uparrow\downarrow}(X, Y, A \circ B) \subseteq \mathrm{Ext}^{\uparrow\downarrow}(X, F, A)$.

Definition 1. *An* **L**-*relation* $\beta \in L^{X_1 \times Y_2}$ *is called a* homogeneous weak **L**-bond[3] *from* **L**-*context* $\langle X_1, Y_1, I_1 \rangle$ *to* **L**-*context* $\langle X_2, Y_2, I_2 \rangle$ *if*

$$\begin{aligned} \mathrm{Ext}^{\uparrow\downarrow}(X_1, Y_2, \beta) &\subseteq \mathrm{Ext}^{\uparrow\downarrow}(X_1, Y_1, I_1), \\ \mathrm{Int}^{\uparrow\downarrow}(X_1, Y_2, \beta) &\subseteq \mathrm{Int}^{\uparrow\downarrow}(X_2, Y_2, I_2). \end{aligned} \tag{9}$$

In this paper we assume only weak homogeneous **L**-bonds w.r.t. $\langle \cap, \cup \rangle$. In what follows, we omit the words 'weak homogeneous' and the pair of concept-forming operators and call them just '**L**-bonds'.

We will utilize the following characterization of **L**-bonds.

Lemma 2 ([7]). *An* **L**-*relation* $\beta \in L^{X_1 \times Y_2}$ *is an* **L**-*bond from* $\langle X_1, Y_1, I_1 \rangle$ *to* $\langle X_2, Y_2, I_2 \rangle$ *iff there is such* **L**-*relation* S_{e} *that* $\beta = S_{\mathrm{e}} \circ I_2$ *and* $\cup_{S_{\mathrm{e}}}$ *maps extents of* $\mathcal{B}^{\uparrow\downarrow}(X_2, Y_2, I_2)$ *to extents of* $\mathcal{B}^{\uparrow\downarrow}(X_2, Y_2, I_2)$.

Remark 2. Note that due to results on fuzzy relational equations we have that the **L**-relation S_{e} from Lemma 2 is equal to $\beta \triangleright I_2^{\mathrm{T}}$ (see [2]).

[3] The notion of **L**-bond was introduced in [12]; however we adapt its definition the same way as in [8, 10] w.r.t. $\langle \cap, \cup \rangle$

3 Results

Firstly, we describe compositions of **L**-bonds and show that they form a category. Later we generalize the results to setting of isotone bonds between fuzzy contexts over different complete residuated lattices.

3.1 Setting with uniform structures of truth degrees

We start with the notion of composition of **L**-bonds.

Definition 2. *Let β_1 be an* **L***-bond from $\langle X_1, Y_1, I_1\rangle$ to $\langle X_2, Y_2, I_2\rangle$ and β_2 be an* **L***-bond from $\langle X_2, Y_2, I_2\rangle$ to $\langle X_3, Y_3, I_3\rangle$. Define composition of β_1 and β_2 as the* **L***-relation $(\beta_1 \triangleright I_2^{\mathrm{T}}) \circ \beta_2 \in L^{X_1 \times Y_3}$ and denote it $\beta_1 \bullet \beta_2$.*

Theorem 2. *The composition of* **L***-bonds is an* **L***-bond.*

Proof. Let β_1 be an **L**-bond from $\langle X_1, Y_1, I_1\rangle$ to $\langle X_2, Y_2, I_2\rangle$ and β_2 be an **L**-bond from $\langle X_2, Y_2, I_2\rangle$ to $\langle X_3, Y_3, I_3\rangle$. By Lemma 2 there are $S_{\mathrm{e}} \in L^{X_1 \times X_2}, {S_{\mathrm{e}}}' \in L^{X_2 \times X_3}$ such that $\beta_1 = S_{\mathrm{e}} \circ I_2$, $\beta_2 = {S_{\mathrm{e}}}' \circ I_3$. By Definition 2 and Remark 2 we have

$$\begin{aligned}\beta_1 \bullet \beta_2 &= (\beta_1 \triangleright I_2^{\mathrm{T}}) \circ \beta_2 \\ &= S_{\mathrm{e}} \circ {S_{\mathrm{e}}}' \circ I_3.\end{aligned}$$

Hence we have

$$\mathrm{Int}^{\cap\cup}(X_1, Y_3, \beta_1 \bullet \beta_2) \subseteq \mathrm{Int}^{\cap\cup}(X_3, Y_3, I_3) \tag{10}$$

by Lemma 1 (a). Note that the mapping $\cup_{S_{\mathrm{e}}}$ maps extents of I_2 to extents of I_1 by Lemma 2 and that $B^{\cup_{\beta_2}}$ is extent of I_2 for any $B \in \mathrm{Int}^{\cap\cup}(X_3, Y_3, I_3)$ by (8). Thus we have

$$B^{\cup_{\beta_1 \bullet \beta_2}} = B^{\cup_{\beta_2} \cup_{S_{\mathrm{e}}}} \in \mathrm{Ext}^{\cap\cup}(X_1, Y_1, I_1),$$

hence

$$\mathrm{Ext}^{\cap\cup}(X_1, Y_3, \beta_1 \bullet \beta_2) \subseteq \mathrm{Ext}^{\cap\cup}(X_1, Y_1, I_1). \tag{11}$$

The equalities (10) and (11) imply that $\beta_1 \bullet \beta_2$ is an **L**-bond. □

Lemma 3. *Let β be an* **L***-bond from* **L***-context $\langle X_1, Y_1, I_1\rangle$ to* **L***-context $\langle X_2, Y_2, I_2\rangle$. For any* **L***-set $A \in L^{X_1}$ we have that $A^{\cap_{I_1} \cup_{I_1} \cap_\beta} = A^{\cap_\beta}$.*

Proof. Let A be an arbitrary **L**-set from L^{X_1}. Then

$$\begin{aligned}A^{\cap_{I_1} \cup_{I_1} \cap_\beta} &\supseteq A^{\cap_\beta} \text{ since } (-)^{\cap_\beta} \text{ is isotone and } A^{\cap_{I_1} \cup_{I_1}} \supseteq A \\ &= A^{\cap_\beta \cup_\beta \cap_\beta} \\ &= A^{\cap_\beta \cup_\beta \cap_{I_2} \cup_{I_2} \cap_\beta} \text{ due to definition of } \mathbf{L}\text{-bond} \\ &\supseteq A^{\cap_{I_1} \cup_{I_1} \cap_\beta} \text{ since the mapping } (-)^{\cap_{I_1} \cup_{I_1} \cap_\beta} \text{ is isotone}\end{aligned}$$

Hence $A^{\cap_{I_1} \cup_{I_1} \cap_\beta} = A^{\cap_\beta}$. □

The equality from Lemma 3 written in relational form is $A \circ \beta = (A \circ I_1) \triangleright I_1^{\mathrm{T}}) \circ \beta$; we use that to prove the following theorem.

Theorem 3. *Composition of* **L***-bonds is associative.*

Proof. Let β_1 be an **L**-bond from $\langle X_1, Y_1, I_1 \rangle$ to $\langle X_2, Y_2, I_2 \rangle$, β_2 be an **L**-bond from $\langle X_2, Y_2, I_2 \rangle$ to $\langle X_3, Y_3, I_3 \rangle$, and β_3 be an **L**-bond from $\langle X_3, Y_3, I_3 \rangle$ to $\langle X_4, Y_4, I_4 \rangle$. We have

$$
\begin{aligned}
(\beta_1 \bullet \beta_2) \bullet \beta_3 &= (((\beta_1 \triangleright I_2^{\mathrm{T}}) \circ \beta_2) \triangleright I_3^{\mathrm{T}}) \circ \beta_3 && \text{by Definition 2} \\
&= ((S_{\mathrm{e}} \circ \beta_2) \triangleright I_3^{\mathrm{T}}) \circ \beta_3 && \text{by Remark 2} \\
&= ((S_{\mathrm{e}} \circ (S'_e \circ I_3)) \triangleright I_3^{\mathrm{T}}) \circ \beta_3 && \text{by Lemma 2} \\
&= (((S_{\mathrm{e}} \circ S'_e) \circ I_3) \triangleright I_3^{\mathrm{T}}) \circ \beta_3 && \text{by (3)} \\
&= (S_{\mathrm{e}} \circ S'_e) \circ \beta_3 && \text{by Lemma 3} \\
&= S_{\mathrm{e}} \circ (S'_e \circ \beta_3) && \text{by (3)} \\
&= S_{\mathrm{e}} \circ (\beta_2 \bullet \beta_3) = \beta_1 \bullet (\beta_2 \bullet \beta_3) && \text{by Remark 2 and Definition 2.}
\end{aligned}
$$

□

We obtain a category of **L**-contexts and **L**-bonds.

Theorem 4. *The structure of* **L***-contexts and* **L***-bonds forms a category:*

Objects *are* **L***-contexts,*
Arrows *are* **L***-bonds where*
identity arrow *of any formal* **L***-context* $\langle X, Y, I \rangle$ *is its incidence relation* I*,*[4]
composition of arrows $\beta_1 \bullet \beta_2$ *is given by Definition 2.*

Remark 3. The category is equivalent to category of attribute-oriented concept lattices and isotone Galois connections. That is analogous to results in [12]. We will bring more about is in full version of the paper.

3.2 Setting with different structures of truth degrees

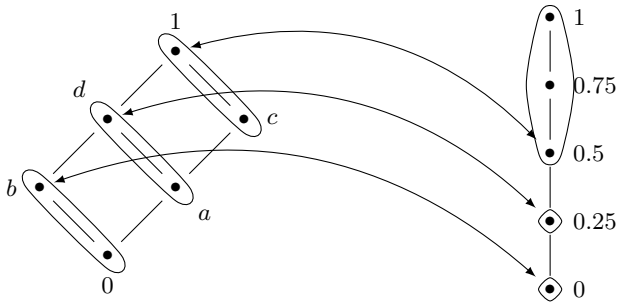
In this section we generalize the previous results into a setting in which fuzzy contexts are defined over different complete residuated lattices. To do that we need to explore compositions of underlying morphisms called residuation-preserving (l, k)-connections between complete residuated lattices.

(l, k)-connections and their compositions

Firstly, let us recall definition and basic properties of the (l, k)-connections introduced in [9].

Definition 3 ([9]). *Let* $\mathbf{L}_1, \mathbf{L}_2$ *be complete residuated lattices, let* $l \in L_1, k \in L_2$ *and let* $\lambda : L_1 \to L_2, \kappa : L_2 \to L_1$ *be mappings, such that*

[4] Clearly, I is an **L**-bond from $\langle X, Y, I \rangle$ to $\langle X, Y, I \rangle$.



$\otimes$	0	a	b	c	d	1
0	0	0	0	0	0	0
a	0	0	0	a	0	a
b	0	0	b	0	b	b
c	0	a	0	c	a	c
d	0	0	b	a	b	d
1	0	a	b	c	d	1

$\rightarrow$	0	a	b	c	d	1
0	1	1	1	1	1	1
a	d	1	d	1	1	1
b	c	c	1	c	1	1
c	b	d	b	1	d	1
d	a	c	d	c	1	1
1	0	a	b	c	d	1

Fig. 1. Six-element residuated lattice, with $\otimes$ and $\rightarrow$ as showed in the bottom part (`011010:00A0B0BCAB` in [3]), (top left), five-element Łukasiewicz chain (`111:000AB` in [3]), (top right), and $(c, 0.5)$-connection between them.

- $\langle \lambda, \kappa \rangle$ *is an isotone Galois connection between* $\mathbf{L}_1$ *and* $\mathbf{L}_2$,
- $\kappa\lambda(a_1) = l \rightarrow_1 (l \otimes_1 a_1)$ *for each* $a_1 \in L_1$,
- $\lambda\kappa(a_2) = k \otimes_2 (k \rightarrow_2 a_2)$ *for each* $a_2 \in L_2$.

We call $\langle \lambda, \kappa \rangle$ *an* (l, k)-connection *from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$. *An* (l, k)-connection *from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$ *is called* residuation-preserving *if*

$$\kappa(k \otimes_2 (\lambda(a) \rightarrow_2 \lambda(b))) = \kappa\lambda(a) \rightarrow_1 \kappa\lambda(b) \tag{12}$$

holds true for any $a, b \in L_2$.

Theorem 5 ([9]). *Let* $\langle \lambda, \kappa \rangle$ *be a residuation-preserving* (l, k)*-connection from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$. *The algebra* $\langle \mathrm{fix}(\lambda, \kappa), \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$ *where* $\wedge$ *and* $\vee$ *are given by the order*

$$\langle a_1, a_2 \rangle \leqslant \langle b_1, b_2 \rangle \quad \begin{array}{l} \text{if } a_1 \leqslant_1 b_1, \\ \text{(equivalently, if } a_2 \leqslant_2 b_2\text{)} \end{array} \tag{13}$$

and the adjoint pair is given by

$$\langle a_1, a_2 \rangle \rightarrow \langle b_1, b_2 \rangle = \langle a_1 \rightarrow_1 b_1, k \otimes_2 (a_2 \rightarrow_2 b_2) \rangle \tag{14}$$

$$= \langle a_1 \rightarrow_1 b_1, k \otimes_2 ((k \rightarrow_2 a_2) \rightarrow_2 (k \rightarrow_2 b_2)) \rangle, \tag{15}$$

$$\langle a_1, a_2 \rangle \otimes \langle b_1, b_2 \rangle = \langle l \rightarrow_1 (l \otimes_1 a_1 \otimes_1 b_1), a_2 \otimes_2 (k \rightarrow_2 b_2) \rangle \tag{16}$$

$$= \langle l \rightarrow_1 (l \otimes_1 a_1 \otimes_1 b_1), (k \rightarrow_2 a_2) \otimes_2 b_2 \rangle \tag{17}$$

is a complete residuated lattice.

Figure 1 shows an example of (l,k)-connection. We refer the reader to [9] for ideas behind (l,k)-connections, examples and further details.

Now we define composition of (l,k)-connections and show that it is an (l,k)-connection as well. In addition, the composition preserves residuation-preservation, that means that composition of residuation-preserving (l,k)-connections is a residuation-preserving (l,k)-connection as well.

Theorem 6. *Let $\langle \lambda_1, \kappa_1 \rangle$ be an (l_1, k_2)-connection from $\mathbf{L}_1$ to $\mathbf{L}_2$ and $\langle \lambda_2, \kappa_2 \rangle$ be an (k_2, j_3)-connection from $\mathbf{L}_2$ to $\mathbf{L}_3$. Then the pair of mappings $\lambda\colon \mathbf{L}_1 \to \mathbf{L}_3$, $\kappa\colon \mathbf{L}_3 \to \mathbf{L}_1$, defined by*

$$\begin{aligned} \lambda(a_1) &= \lambda_2(k_2 \to_2 \lambda_1(a_1)), \\ \kappa(a_3) &= \kappa_1(k_2 \otimes_2 \kappa_2(a_3)) \end{aligned} \tag{18}$$

for each $a_1 \in L_1$ and $a_2 \in L_2$, is an (l_1, j_3)-connection from $\mathbf{L}_1$ to $\mathbf{L}_3$.

Proof. First, we prove that $\kappa\lambda(a_1) = l_1 \to_1 (l_1 \otimes_1 a_1)$ for each $a_1 \in L_1$ and $\lambda\kappa(a_2) = j_3 \otimes_3 (j_3 \to_3 a_3)$ for each $a_3 \in L_3$. For each $a_1 \in \mathbf{L}_1$, we have

$$\begin{aligned} \kappa\lambda(a_1) &= \kappa_1(k_2 \otimes_2 \kappa_2(\lambda_2(k_2 \to_2 \lambda_1(a_1)))) \\ &= \kappa_1(k_2 \otimes_2 (k_2 \to_2 (k_2 \otimes (k_2 \to_2 \lambda_1(a_1))))) \\ &= \kappa_1(k_2 \otimes_2 (k_2 \to_2 \lambda_1(a_1))) \\ &= \kappa_1(\lambda_1(\kappa_1(\lambda_1(a_1)))) \\ &= \kappa_1(\lambda_1(a_1)) \\ &= l_1 \to_1 (l_1 \otimes_1 a_1). \end{aligned}$$

Similarly, we have for each $a_3 \in L_3$

$$\begin{aligned} \lambda\kappa(a_3) &= \lambda_2(k_2 \to_2 \lambda_1(\kappa_1(k_2 \otimes_2 \kappa_2(a_3)))) \\ &= \lambda_2(k_2 \to_2 (k_2 \otimes_2 (k_2 \to_2 (k_2 \otimes_2 \kappa_2(a_3))))) \\ &= \lambda_2(k_2 \to_2 (k_2 \otimes_2 \kappa_2(a_3)))) \\ &= \lambda_2(\kappa_2(\lambda_2(\kappa_2(a_3)))) \\ &= \lambda_2(\kappa_2(a_3)) \\ &= j_3 \otimes_3 (j_3 \to_3 a_3). \end{aligned}$$

Since $\kappa\lambda(a_1) = l_1 \to_1 (l_1 \otimes_1 a_1) \geqslant_1 a_1$ and $\lambda\kappa(a_3) = j_3 \otimes_3 (j_3 \otimes_3 a_3) \leqslant_3 a_3$ we only need to show monotony to prove that $\langle \lambda, \kappa \rangle$ is an isotone Galois connection: For each $a_1, b_1 \in \mathbf{L}_1$ we have

$$\begin{aligned} a_1 \leqslant_1 b_1 &\text{ implies } \lambda_1(a_1) \leqslant_2 \lambda_1(b_1) \text{ since } \lambda_1 \text{ is monotone,} \\ &\text{implies } k_2 \to_2 \lambda_1(a_1) \leqslant_2 k_2 \to_2 \lambda_1(b_1) \text{ since } \to_2 \text{ is monotone} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{in its second argument,} \\ &\text{implies } \lambda_2(k_2 \to_2 \lambda_1(a_1)) \leqslant_3 \lambda_2(k_2 \to_2 \lambda_1(b_1)) \text{ since } \lambda_2 \text{ is monotone.} \end{aligned}$$

Thus $a_1 \leqslant_1 b_1$ implies $\lambda(a_1) \leqslant_3 \lambda(b_1)$ for each $a_1, b_1 \in \mathbf{L}_1$. Similarly, one can show that $a_3 \leqslant_3 b_3$ implies $\kappa(a_3) \leqslant_1 \kappa(b_3)$.

□

Theorem 7. *Let $\langle \lambda_1, \kappa_1 \rangle$ be a residuation-preserving (l_1, k_2)-connection from $\mathbf{L}_1$ to $\mathbf{L}_2$ and $\langle \lambda_2, \kappa_2 \rangle$ be a residuation-preserving (k_2, j_3)-connection from $\mathbf{L}_2$ to $\mathbf{L}_3$. Then the pair of mappings $\lambda\colon \mathbf{L}_1 \to \mathbf{L}_3$, $\kappa\colon \mathbf{L}_3 \to \mathbf{L}_1$, defined by* (18)*, is a residuation-preserving (l_1, j_3)-connection from $\mathbf{L}_1$ to $\mathbf{L}_3$.*

Proof. For each $a_1, b_1 \in \mathbf{L}_1$ we have

$$
\begin{aligned}
&\kappa\lambda(a_1) \to_1 \kappa\lambda(b_1) = \\
&= (l_1 \to_1 (l_1 \otimes_1 a_1)) \to_1 (l_1 \to_1 (l_1 \otimes_1 b_1)) \\
&= \kappa_1\lambda_1(a_1) \to_1 \kappa_1\lambda_1(b_1) \\
&= \kappa_1(k_2 \otimes_2 (\lambda_1(a_1) \to_2 \lambda_1(b_1))) \\
&= \kappa_1(k_2 \otimes_2 (\lambda_1\kappa_1\lambda_1(a_1) \to_2 \lambda_1\kappa_1\lambda_1(b_1))) \\
&= \kappa_1(k_2 \otimes_2 ((k_2 \otimes_2 (k_2 \to_2 \lambda_1(a_1))) \to_2 (k_2 \otimes_2 (k_2 \to_2 \lambda_1(b_1))))) \\
&= \kappa_1(k_2 \otimes_2 ((k_2 \otimes_2 (k_2 \to_2 (k_2 \otimes_2 (k_2 \to_2 \lambda_1(a_1))))) \to_2 (k_2 \otimes_2 (k_2 \to_2 \lambda_1(b_1))))) \\
&= \kappa_1(k_2 \otimes_2 ((k_2 \to_2 (k_2 \otimes_2 (k_2 \to_2 \lambda_1(a_1)))) \to_2 (k_2 \to_2 (k_2 \otimes_2 (k_2 \to_2 \lambda_1(b_1)))))) \\
&= \kappa_1(k_2 \otimes_2 (\kappa_2\lambda_2(k_2 \to_2 \lambda_1(a_1)) \to_2 \kappa_2\lambda_2(k_2 \to_2 \lambda_1(b_1)))) \\
&= \kappa_1(k_2 \otimes_2 \kappa_2(j_3 \otimes_3 (\lambda_2(k_2 \to_2 \lambda_1(a_1)) \to_3 \lambda_2(k_2 \to_2 \lambda_1(b_1))))) \\
&= \kappa_1(k_2 \otimes_2 \kappa_2(j_3 \otimes_3 (\lambda(a_1) \to_3 \lambda(b_1)))) \\
&= \kappa(j_3 \otimes_3 (\lambda(a_1) \to_3 \lambda(b_1))).
\end{aligned}
$$

□

We call $\langle \lambda, \kappa \rangle$ from (18) a composition of $\langle \lambda_1, \kappa_1 \rangle$ and $\langle \lambda_2, \kappa_2 \rangle$ and we denote it as $\langle \lambda_1, \kappa_1 \rangle \bullet \langle \lambda_2, \kappa_2 \rangle = \langle \lambda_1 \bullet \lambda_2, \kappa_1 \bullet \kappa_2 \rangle$. Now we show, that the composition of (l, k)-connections is associative.

Theorem 8. *Let $\langle \lambda_1, \kappa_1 \rangle$ be an (l_1, k_2)-connection from $\mathbf{L}_1$ to $\mathbf{L}_2$, $\langle \lambda_2, \kappa_2 \rangle$ be a (k_2, j_3)-connection from $\mathbf{L}_2$ to $\mathbf{L}_3$, and $\langle \lambda_3, \kappa_3 \rangle$ be a (j_3, m_4)-connection from $\mathbf{L}_3$ to $\mathbf{L}_4$. Then*

$$\langle \lambda_1, \kappa_1 \rangle \bullet (\langle \lambda_2, \kappa_2 \rangle \bullet \langle \lambda_3, \kappa_3 \rangle) = (\langle \lambda_1, \kappa_1 \rangle \bullet \langle \lambda_2, \kappa_2 \rangle) \bullet \langle \lambda_3, \kappa_3 \rangle.$$

Proof. We have for each $a \in L_1$

$$
\begin{aligned}
(\lambda_1 \bullet (\lambda_2 \bullet \lambda_3))(a_1) &= (\lambda_2 \bullet \lambda_3)(k_2 \to_2 \lambda_1(a_1)) \\
&= \lambda_3(j_3 \to \lambda_2(k_2 \to_2 \lambda_1(a_1))) \\
&= \lambda_3(j_3 \to (\lambda_1 \bullet \lambda_2)(a_1)) \\
&= ((\lambda_1 \bullet \lambda_2) \bullet \lambda_3)(a_1)
\end{aligned}
$$

and similarly for the κ-part. □

Theorem 9. *The following structure forms a category.*

Objects *are pairs $\langle \mathbf{L}, e \rangle$, where $\mathbf{L}$ is a complete residuated lattices and $e \in L$.*
Arrows *from $\langle \mathbf{L}_1, l \rangle$ to $\langle \mathbf{L}_2, k \rangle$ are (l, k)-connections from $\mathbf{L}_1$ to $\mathbf{L}_2$, where*

identity arrow *on any* $\langle \mathbf{L}, e\rangle$ *is* (e,e)*-connection* $\langle \lambda, \kappa\rangle$ *where* $\lambda(a) = e \otimes a$ *and* $\kappa(a) = e \to a$ *for each* $a \in L$.
composition of arrows *is as defined in* (18).

If we use just residuation-preserving (l,k)*-connections we obtain a sub-category.*

Now, we can explore bonds based on residuation-preserving (l,k)-connections.

Definition 4. *Let* $\mathbf{L}_1, \mathbf{L}_2$ *be complete residuated lattices,* $\langle \lambda, \kappa\rangle$ *be residuation-preserving* (l,k)*-connection from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$*, and let* $\langle X_1, Y_1, I_1\rangle$ *and* $\langle X_2, Y_2, I_2\rangle$ *be* $\mathbf{L}_1$*-context and* $\mathbf{L}_2$*-context, respectively. We call* $\beta \in L_{\langle\lambda,\kappa\rangle}^{X_1 \times Y_2}$ *a* $\langle \lambda, \kappa\rangle$-bond *from* $\langle X_1, Y_1, I_1\rangle$ *to* $\langle X_2, Y_2, I_2\rangle$ *if the following inclusions hold.*

$$\mathrm{Ext}^{\triangle\triangledown}(X_1, Y_2, \beta) \subseteq \mathrm{Ext}^{\cap\cup}(X_1, Y_1, \kappa\lambda(I_1)), \tag{19}$$

$$\mathrm{Int}^{\triangle\triangledown}(X_1, Y_2, \beta) \subseteq \mathrm{Int}^{\cap\cup}(X_2, Y_2, \lambda\kappa(I_2)). \tag{20}$$

The concept-forming operators $\langle \triangle, \triangledown\rangle$ *induced by* $\langle \lambda, \kappa\rangle$*-bond* β *from* $\langle X_1, Y_1, I_1\rangle$ *to* $\langle X_2, Y_2, I_2\rangle$ *are given by*[5]

$$\begin{aligned} A^{\triangle_\beta} &= \lambda(A)^{\cap_{\mathrm{proj}_2(\beta)}}, \\ B^{\triangledown_\beta} &= \kappa(B)^{\cup_{\mathrm{proj}_1(\beta)}}. \end{aligned} \tag{21}$$

Theorem 10. *Let* $\langle X_1, Y_1, I_1\rangle$ *be an* $\mathbf{L}_1$*-context,* $\langle X_2, Y_2, I_2\rangle$ *be an* $\mathbf{L}_2$*-context, and* $\langle \lambda, \kappa\rangle$ *an* (l,k)*-connection from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$*. Then* $\beta \in L_{\langle\lambda,\kappa\rangle}$ *is a* $\langle \lambda, \kappa\rangle$*-bond from* $\langle X_1, Y_1, I_1\rangle$ *to* $\langle X_2, Y_2, I_2\rangle$ *if and only if it is a* $\mathbf{L}_{\langle\lambda,\kappa\rangle}$*-bond w.r.t.* $\langle \cap, \cup\rangle$ *from* $\langle X_1, Y_1, \langle \kappa\lambda(I_1), \lambda(I_1)\rangle\rangle$ *to* $\langle X_2, Y_2, \langle \kappa(I_2), \lambda\kappa(I_2)\rangle\rangle$.

Proof. Directly from the definition and (21). □

For what follows we will need the following product of fuzzy relations. Let $\langle \lambda_1, \kappa_1\rangle$ be (l_1, k_2)-connection from $\mathbf{L}_1$ to $\mathbf{L}_2$, $\langle \lambda_2, \kappa_2\rangle$ be (k_2, m_3)-connection from $\mathbf{L}_2$ to $\mathbf{L}_3$, and $I \in \mathbf{L}_{\langle\lambda_1,\kappa_1\rangle}^{X\times Y}$, $J \in \mathbf{L}_{\langle\lambda_2,\kappa_2\rangle}^{Y\times Z}$. Then $I \boxtimes J \in \mathbf{L}_{\langle\lambda_1\bullet\lambda_2, \kappa_1\bullet\kappa_2\rangle}^{X\times Z}$ is defined as

$$I \boxtimes J = \langle \kappa_1(K), \lambda_2(k_2 \to_2 K)\rangle \quad \text{where } K = \mathrm{proj}_2(I) \circ_2 \mathrm{proj}_1(J) \tag{22}$$

and $\circ_2$ is composition of $\mathbf{L}_2$-relations (1).

Lemma 4. *Let* $\langle X_1, Y_1, I_1\rangle$ *be an* $\mathbf{L}_1$*-context,* $\langle X_2, Y_2, I_2\rangle$ *be an* $\mathbf{L}_2$*-context, and* $\langle \lambda, \kappa\rangle$ *an* (l,k)*-connection from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$.

(a) An $\mathbf{L}_{\langle\lambda,\kappa\rangle}$*-relation* β *for which exist* $\mathbf{L}_{\langle\lambda,\kappa\rangle}$*-relations* $S_\mathrm{e} \in L_{\langle\lambda,\kappa\rangle}^{X_1\times X_2}$ *and* $S_\mathrm{i} \in L_{\langle\lambda,\kappa\rangle}^{Y_1\times Y_2}$ *such that*

$$\beta = \langle \kappa\lambda(I_1), \lambda(I_1)\rangle \boxtimes S_\mathrm{i} = S_\mathrm{e} \boxtimes \langle \kappa(I_2), \lambda\kappa(I_2)\rangle$$

is a $\langle \lambda, \kappa\rangle$*-bond from* $\langle X_1, Y_1, I_1\rangle$ *to* $\langle X_2, Y_2, I_2\rangle$.

[5] proj_1, proj_2 denote projection of first and second component of a pair, respectively.

(b) Each $\langle \lambda, \kappa \rangle$*-bond* β *from* $\langle X_1, Y_1, I_1 \rangle$ *to* $\langle X_2, Y_2, I_2 \rangle$ *satisfies that there is* $S_{\mathrm{e}} \in L_{\langle \lambda,\kappa \rangle}^{X_1 \times X_2}$ *such that*

$$\beta = S_{\mathrm{e}} \boxtimes \langle \kappa(I_2), \lambda\kappa(I_2) \rangle.$$

Proof. From Theorem 10 and Lemma 1. □

Theorem 11. *Let* $\langle \lambda_1, \kappa_1 \rangle$ *be an* (l_1, k_2)*-connection from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$*,* $\langle \lambda_2, \kappa_2 \rangle$ *be an* (k_2, j_3)*-connection from* $\mathbf{L}_2$ *to* $\mathbf{L}_3$*,* β_1 *be* $\langle \lambda_1, \kappa_1 \rangle$*-bond from* $\langle X_1, Y_1, I_1 \rangle$ *to* $\langle X_2, Y_2, I_2 \rangle$*, and* β_2 *be* $\langle \lambda_2, \kappa_2 \rangle$*-bond from* $\langle X_2, Y_2, I_2 \rangle$ *to* $\langle X_3, Y_3, I_3 \rangle$*.*

$$\beta = S_{\mathrm{e}} \boxtimes \beta_2, \tag{23}$$

where $S_{\mathrm{e}} = \beta_1 \triangleright \langle \kappa_1(I_2^{\mathrm{T}}), \lambda_1\kappa_1(I_2^{\mathrm{T}}) \rangle$*, is a* $\langle \lambda_1 \bullet \lambda_2, \kappa_1 \bullet \kappa_2 \rangle$*-bond from* $\langle X_1, Y_1, I_1 \rangle$ *to* $\langle X_3, Y_3, I_3 \rangle$*.*

Let us denote β from (23) as $\beta = \beta_1 \bullet \beta_2$ and call it a composition of isotone $\langle \lambda, \kappa \rangle$-bonds. Now we show associativity of this composition.

Theorem 12. *Let* $\langle \lambda_1, \kappa_1 \rangle$ *be an* (l_1, k_2)*-connection from* $\mathbf{L}_1$ *to* $\mathbf{L}_2$*,* $\langle \lambda_2, \kappa_2 \rangle$ *be an* (k_2, j_3)*-connection from* $\mathbf{L}_2$ *to* $\mathbf{L}_3$*,* $\langle \lambda_3, \kappa_3 \rangle$ *be an* (j_3, m_4)*-connection from* $\mathbf{L}_3$ *to* $\mathbf{L}_4$*, and* β_i *be* $\langle \lambda_i, \kappa_i \rangle$*-bond from* $\langle X_i, Y_i, I_i \rangle$ *to* $\langle X_{i+1}, Y_{i+1}, I_{i+1} \rangle$*. Then*

$$\beta_1 \bullet (\beta_2 \bullet \beta_3) = (\beta_1 \bullet \beta_2) \bullet \beta_3.$$

Proof. Follows from Theorem 3, Theorem 8, and Theorem 10. □

Finally, we can state that **L**-contexts over different structures of truth degrees and bonds between them form a category.

Theorem 13. Objects *are pairs* $\langle \mathbf{K}, e \rangle$*, where* $\mathbf{K}$ *is a* $\mathbf{L}$*-context and* $e \in L$*.*
Arrows *between* $\langle \mathbf{K_1}, l \rangle$ *and* $\langle \mathbf{K_2}, k \rangle$*, where* $\mathbf{K}_1$ *is an* $\mathbf{L}_1$*-context,* $\mathbf{K}_2$ *is an* $\mathbf{L}_2$*-context and* $l \in L_1, k \in L_2$*, are* $\langle \lambda, \kappa \rangle$*-bonds, where* $\langle \lambda, \kappa \rangle$ *is an* (l, k)*-connection.*
identity arrow *for a pair* $\langle K, e \rangle$ *of* $\mathbf{L}$*-context* $\langle X, Y, I \rangle$ *and* e *is* $\langle \lambda, \kappa \rangle$*-bond* I *with* $\langle \lambda, \kappa \rangle$ *are* (e, e)*-connections* $\langle \lambda, \kappa \rangle$ *where* $\lambda(x) = e \to a$ *and* $\kappa(x) = e \otimes a$ *for each* $a \in L$*.*
composition of arrows $\beta_1 \bullet \beta_2$ *is given by* (23)*.*

4 Future Research

Our future research in this area includes addressing the following issues:

- Antitone bonds between fuzzy contexts over different complete residuated lattices were studied in [9]; basics of Isotone bonds are presented in this paper. We want to extend this study to heterogeneous bonds[11]. We will bring results on them and their compositions in the full version of this paper.
- As block relations are a special case of bonds, they share many properties (see [11]). It can be fruitful to study the compositions described in this paper in context of block **L**-relations. In addition, the composition applied on block (crisp) relations correspond with multiplication used in calculus studied in [1]. This observation deserves deeper study; we believe that this can bring a new interesting insight to the calculus.

Acknowledgments

Jan Konecny is supported by grant No. 15-17899S, "Decompositions of Matrices with Boolean and Ordinal Data: Theory and Algorithms", of the Czech Science Foundation.

Ondrej Krídlo is supported by grant VEGA 1/0073/15 by the Ministry of Education, Science, Research and Sport of the Slovak republic and University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF.

References

1. Eduard Bartl and Michal Krupka. Residuated lattices of block relations: Size reduction of concept lattices. *to appear in IJGS*, 2015.
2. Radim Belohlavek. *Fuzzy Relational Systems: Foundations and Principles*. Kluwer Academic Publishers, Norwell, USA, 2002.
3. Radim Belohlavek and Vilem Vychodil. Residuated lattices of size $\leqslant 12$. *Order*, 27(2):147–161, 2010.
4. Radim Belohlavek and Jan Konecny. Row and Column Spaces of Matrices over Residuated Lattices. *Fundam. Inform.*, 115(4):279–295, 2012.
5. George Georgescu and Andrei Popescu. Non-dual fuzzy connections. *Arch. Math. Log.*, 43(8):1009–1039, 2004.
6. Ladislav J Kohout and Wyllis Bandler. Relational-product architectures for information processing. *Information Sciences*, 37(1-3):25–37, 1985.
7. Jan Konecny. Isotone fuzzy Galois connections with hedges. *Information Sciences*, 181(10):1804–1817, 2011.
8. Jan Konecny. Antitone L-bonds. In: Information Processing and Management of Uncertainty in Knowledge-Based Systems – 15th International Conference, IPMU 2014, pages 71–80, 2014.
9. Jan Konecny. Bonds between L-fuzzy contexts over different structures of truth-degrees, In: 13th International Conference, ICFCA 2015, Nerja, Spain, June 23-26, pages 81–96, 2015.
10. Jan Konecny and Manuel Ojeda-Aciego. Isotone L-bonds. In Manuel Ojeda-Aciego and Jan Outrata, editors, *CLA*, volume 1062 of *CEUR Workshop Proceedings*, pages 153–162. CEUR-WS.org, 2013.
11. Jan Konecny and Manuel Ojeda-Aciego. On homogeneous L-bonds and heterogeneous L-bonds. *to appear in IJGS*, 2015.
12. Ondrej Krídlo, Stanislav Krajči, and Manuel Ojeda-Aciego. The Category of L-Chu Correspondences and the Structure of L-Bonds. *Fundam. Inform.*, 115(4):297–325, 2012.
13. Ondrej Kridlo and Manuel Ojeda-Aciego. CRL-Chu Correspondences. In Manuel Ojeda-Aciego and Jan Outrata, editors, *CLA*, volume 1062 of *CEUR Workshop Proceedings*, pages 105–116. CEUR-WS.org, 2013.
14. Markus Krötzsch, Pascal Hitzler, and Guo-Qiang Zhang. Morphisms in context. In *Conceptual Structures: Common Semantics for Sharing Knowledge, 13th International Conference on Conceptual Structures, ICCS 2005, Kassel, Germany, July 17-22, 2005, Proceedings*, pages 223–237, 2005.
15. Jesús Medina. Multi-adjoint property-oriented and object-oriented concept lattices. *Inf. Sci.*, 190:95–106, 2012.

From an implicational system to its corresponding *D*-basis

Estrella Rodríguez-Lorenzo[1], Kira Adaricheva[2], Pablo Cordero[1], Manuel Enciso[1], and Angel Mora[1]

[1] University of Málaga, Andalucía Tech, Spain,
e-mail: {estrellarodlor,amora}@ctima.uma.es, pcordero@uma.es, enciso@lcc.uma.es
[2] Nazarbayev University, Kazakhstan
e-mail: kira.adaricheva@nu.edu.kz

Abstract. Closure system is a fundamental concept appearing in several areas such as databases, formal concept analysis, artificial intelligence, etc. It is well-known that there exists a connection between a closure operator on a set and the lattice of its closed sets. Furthermore, the closure system can be replaced by a set of implications but this set has usually a lot of redundancy inducing non desired properties.
In the literature, there is a common interest in the search of the minimality of a set of implications because of the importance of bases. The well-known Duquenne-Guigues basis satisfies this minimality condition. However, several authors emphasize the relevance of the optimality in order to reduce the size of implications in the basis. In addition to this, some bases have been defined to improve the computation of closures relying on the directness property. The efficiency of computation with the direct basis is achieved due to the fact that the closure is computed in one traversal.
In this work, we focus on the *D*-basis, which is ordered-direct. An open problem is to obtain it from an arbitrary implicational system, so it is our aim in this paper. We introduce a method to compute the *D*-basis by means of minimal generators calculated using the Simplification Logic for implications.

1 Introduction

Discovering knowledge and information retrieval are currently active issues where Formal Concept Analysis (FCA) provides tools and methods for data analysis. The notions around the concept lattice may be considered as the main attractions in Formal Concept Analysis and they are strongly connected to the notion of closure.

Closure system is a fundamental concept appearing in several areas such as database theory, formal concept analysis, artificial intelligence, etc. It is well-known that there exists a connection between a closure operator on a set and the lattice of its closed sets. Furthermore, the closure system can be presented, dually, as a set of attribute implications, namely an implicational system but this set has usually a lot of redundancy inducing non-desired properties.

We can not fail to mention the relevance of the role of the implication notion in different areas. It was the main actor of the normalization theory in database area; it has an outstanding character in Formal Concept Analysis and it was prominently used in Frequent Set Mining and Learning Spaces, see the survey of M. Wild [10]. The latter is devoted to mathematical theory of implications and the different faces of the concept of an implication. Implications linked data represented in several forms going from the relationship between itemsets in transactions (Frequent Set Mining) to the boolean functions (Horn Theory).

Nonetheless, as V. Duquenne says in [6] "it is surprising if not hard to acknowledge that we did not learn much more on *their intimacy* in the meantime, despite many interesting papers using or revisiting them". We believe there is a long way to go, and a deeper theory on properties of implications with automated and efficient methods to manipulate them can be developed.

In this paper, we are focused in the Formal Concept Analysis area and the fundamental notions are assumed (see [7]). The task of information retrieval carried out by the tools in FCA conduits to infer *concepts* from the data set, i.e. to deduce (in an automated way) a set of objects that may be precisely characterized by a set of attributes. Such concepts inherit an order relation induced by attribute set inclusion, providing a lattice structure of the concept set. Here implications are retrieved from a binary table (formal context) representing the relationship between a set of objects and a set of attributes. Implications represent an alternative way for the underlying information contained in the formal context.

Many applications must massively compute closures of sets of attributes and any improvement of execution time is relevant. In [9] the author establishes the necessity of obtaining succinct representation of closure operators to achieve an efficient computational usage. In this direction, properties associated to implications are studied to render equivalent sets fulfilling desired properties, *directness* and *optimality*.

An important matter in FCA is to transform implicational systems in canonical forms for special proposals in order to provide an efficient further management. Hence, some alternative definitions have been established: Duquenne-Guigues basis, direct optimal basis, D-basis, etc. In this work we focus on the last one [1], because it combines, in a balanced way, a brief representation (it has a small number of elements) and a efficient computation of closures (it is computed in just one traversal). To this end, D-basis proposes an order in which implications will be attended.

The major issue is that the execution of the D-basis in one iteration is more efficient that the execution of a shorter, but un-ordered one, for instance the *canonical basis* of Duquenne and Guigues. K. Adaricheva et.al prove in [1] that one can extract the D-basis from any direct unit basis Σ in time polynomial of size of Σ, and it takes only linear time of the number of implications of the D-basis to put it into a proper order.

In [5] we have proposed a method to calculate all the minimal generators from a set of implications as a way to remove redundancy in the basis. The method

to compute all the minimal generators is based on the Simplification Logic for implications [8]. Using this logic we are able to remove redundancy in the implications [4] and following the same style of application of the Simplification Rule to the set of implications we can obtain all the minimal generators.

Currently the retrieval of the D-basis from an arbitrary implicational system is an open problem, so it becomes our aim in this paper. We introduce a method to compute the D-basis by means of minimal generators calculated using the Simplification Logic for implications. The relationship among minimal generators, covers, minimal covers and D-basis is presented and an algorithm to calculate D-basis from an arbitrary set of implications is proposed.

Section 2 presents the main notions necessary to the understanding of the new method: closure operators, the D-basis, Simplification Logic and the method to calculate minimal generators. In Section 3, the relationships between covers and generators are presented. In Section 4, the new method to obtain the D-basis from a set of implications is shown, and some conclusions and extensions are proposed in Section 5.

2 Background

2.1 Closure systems

Given a non-empty set M and the set[1] 2^M of all its subsets, *a closure operator* is a map $\phi : 2^M \to 2^M$ that satisfies the following, for all $X, Y \in 2^M$:

(1) increasing: $X \subseteq \phi(X)$;
(2) isotone: $X \subseteq Y$ implies $\phi(X) \subseteq \phi(Y)$;
(3) idempotent: $\phi(\phi(X)) = \phi(X)$.

We will refer to the pair $\langle M, \phi \rangle$ of a set M and a closure operator on it as a *closure system*.

In the next two subsections we will follow the introduction of the implicational system based on the *minimal proper covers*[2] given in [1], which was named there the D-basis.

We will call closure system *reduced*, if $\phi(\{x\}) = \phi(\{y\}) \rightarrow x = y$, for any $x, y \in M$ [3]. If the closure system $\langle M, \phi \rangle$ is not reduced, one can modify it to produce an equivalent one that is reduced, see [1] for more details.

We will now define a closure operator ϕ^*, which is associated with a given operator ϕ.

Definition 1. *Let $\langle M, \phi \rangle$ be a closure system. Define ϕ^* as a self-map on 2^M such that $\phi^*(X) = \bigcup_{x \in X} \phi(x), X \subseteq 2^M$.*

It is straightforward to verify that

[1] In the FCA framework, that set M can be thought a set of attributes of a context.
[2] Although in [1] it was introduced as minimal cover, here we name it minimal proper cover because in this paper we generalize the notion of cover in Section 3.
[3] To clarify the notation $\phi(\{x\})$ will be represented as $\phi(x)$ if no risk of confusion.

Lemma 1. *ϕ^* is a closure operator on M.*

Given a closure system $\langle M, \phi \rangle$, we introduce several important concepts.

Definition 2 ([1]). *For $x \in M$ we call a subset $X \subseteq M$ a proper cover for x if $x \in \phi(X) \setminus \phi^*(X)$. If X is a proper cover for x, it will be denoted as $x \sim_p X$.*

2.2 The *D*-basis

In this subsection, we briefly summarize the introduction of the D-basis in [1]. Its definition is strongly based on the notion of a minimal proper cover:

Definition 3. *A proper cover Y for x is called minimal, if, for any other proper cover Z for x, $Z \subseteq \phi^*(Y)$ implies $Y \subseteq Z$.*

The existence of several proper covers for the same element induces the need to introduce the notion of minimality.

Lemma 2. *If $x \sim_p X$, then there exists Y such that $x \sim_p Y$, $Y \subseteq \phi^*(X)$ and Y is a minimal proper cover for x. In other words, every proper cover can be reduced to a minimal proper cover under the subset relation added with the ϕ^* operator.*

These ideas bring to the following definition of the implicational system defining the reduced closure system by means of the minimal proper covers.

Definition 4. *Given a reduced closure system $\langle M, \phi \rangle$, define the D-**basis** Σ_D as a union of two subsets of implications:*

1. *$\{y \to x : x \in \phi(y) \setminus y, y \in M\}$ (such implications are called* binary*);*
2. *$\{X \to x : X$ is a minimal proper cover for $x\}$.*

Note that the D-basis belongs to the family of the *unit* bases, i.e. implicational sets where each implication $A \to b$ has a singleton $b \in M$ as a consequent.

Lemma 3. *Σ_D generates $\langle M, \phi \rangle$.*

2.3 Ordered direct set of implications

Here we recall the notion of the ordered direct basis introduced in [1], which is designed for a quick computation of the closures based on some fixed order of implications. First we recall the definition of the ordered iteration of implications.

Definition 5. *Suppose the set of implications Σ is equipped with some linear order, or equivalently, the implications are indexed as $\Sigma = \{s_1, s_2, \ldots, s_n\}$. Define a mapping $\rho_\Sigma : 2^M \to 2^M$ associated with this ordering as follows. For any $X \subseteq M$, let $X_0 = X$. If X_k is computed and implication s_{k+1} is $A \to B$, then*

$$X_{k+1} = \begin{cases} X_k \cup B, & \text{if } A \subseteq X_k, \\ X_k, & \text{otherwise.} \end{cases}$$

Finally, $\rho_\Sigma(X) = X_n$. We will call ρ_Σ an ordered iteration *of Σ.*

The concept of the ordered iteration is central for the definition of the ordered direct basis. For any given set of implications Σ on set M, by ϕ_Σ we understand the closure operator on M defined by Σ. Equivalently, the fixed points of ϕ_Σ are exactly subsets $X \subseteq M$ which are stable for all implications $A \to B$ in Σ: if $A \subseteq X$, then $B \subseteq X$.

Definition 6. *The set of implications with some linear ordering on it, $\langle \Sigma, < \rangle$, is called an* ordered direct basis*, if, with respect to this ordering, $\phi_\Sigma(X) = \rho_\Sigma(X)$ for all $X \subseteq S$.*

We note that any *direct* basis is ordered direct. By direct basis we understand any set of implications that allows to produce the closure of subsets of the base set while attending each implication only once.

More precisely, if Σ is some set of implications defining the closure system $\langle M, \phi \rangle$, then let $\pi_\Sigma(X) = X \cup \bigcup\{B : A \subseteq X \text{ and } (A \to B) \in \Sigma\}$. In order to obtain $\phi(X)$, for any $X \subseteq M$, one would normally need to repeat several iterations of π_Σ: $\phi(X) = \pi_\Sigma(X) \cup \pi^2_\Sigma(X) \cup \pi^3_\Sigma(X) \ldots$.

The bases for which one can obtain the closure of any set X performing only one iteration of π_Σ, i.e., $\phi(X) = \pi_\Sigma(X)$, are called *direct*. The various direct bases appearing in the literature were surveyed in K. Bertet and B. Monjardet [3]. Important result of their analysis is that in the family of all possible (unit) direct bases for the same closure system, there exists a $\subseteq$-smallest direct basis Σ_{cd}, which was called as *canonical unit direct* basis.

The following result from [1] summarizes the relation between the D-basis Σ_D and Σ_{cd} for a given closure system.

Theorem 1. $\Sigma_D \subseteq \Sigma_{cd}$.

2.4 Minimal Generators via Simplification Logic

In this subsection we summarize how the inference system of Simplification Logic $\mathbf{SL}_{\text{FD}}$ [4, 8] (equivalent to Armstrong's axioms) is used to enumerate all minimal generators. $\mathbf{SL}_{\text{FD}}$ is guided by the idea of simplifying the set of implications by efficiently removing some redundant attributes.

$\mathbf{SL}_{\text{FD}}$ logic considers reflexivity as axiom scheme

$$\texttt{[Ref]} \quad \frac{A \supseteq B}{A \to B}$$

and the following inference rules called fragmentation, composition and simplification respectively.

$$\texttt{[Frag]} \ \frac{A \to B \cup C}{A \to B} \qquad \texttt{[Comp]} \ \frac{A \to B,\ C \to D}{A \cup C \to B \cup D} \qquad \texttt{[Simp]} \ \frac{A \to B,\ C \to D}{A \cup (C \smallsetminus B) \to D}$$

The important matter is that these rules can be considered as equivalence rules which have been used as the core in automated methods for removing redundancies, obtaining minimal keys, or computing closures. In particular, the last method indicated is based on the following results (see [8] for details, proofs and examples):

Theorem 2 ([8]). *Let $A, B \subseteq M$ and Σ be a set of implications. We have $\Sigma \vdash A \to B$ if and only if $\{\emptyset \to A\} \cup \Sigma \vdash \emptyset \to B$.*

In order to compute closures, since $\phi(A)$ is the biggest subset B such that $\Sigma \vdash A \to B$, the formula $\emptyset \to A$ is used as a seed which guides the reasoning to render the closure $\phi(A)$ just by applying the following equivalences where the [`Simp`] inference rule plays a main role.

Proposition 1. *Let A, B and C be subsets of M, then the following equivalences hold:*

- **Eq. I***: If $B \subseteq A$ then $\{\emptyset \to A, B \to C\} \equiv \{\emptyset \to A \cup C\}$.*
- **Eq. II***: If $C \subseteq A$ then $\{\emptyset \to A, B \to C\} \equiv \{\emptyset \to A\}$.*
- **Eq. III***: Otherwise $\{\emptyset \to A, B \to C\} \equiv \{\emptyset \to A, B \setminus A \to C \setminus A\}$.*

Based on the above proposition and Theorem 2, in [5] the method `Cls` is introduced. This method receives as input set $A \subseteq M$ and a set of implications Σ and renders the pair $(\phi(A), \Sigma')$ where Σ' is the set of implications simplified with respect to $\emptyset \to \phi(A)$ (i.e. Σ contracted to $2^{M \setminus \phi(A)}$).

NOTATION: From now on, in order to simplify the examples, elements of M are denoted by numbers from 0 to 9 or lowercase letters and we omit brackets and commas in subsets of M.

Example 1. Let Σ be $\{ab \to c, ac \to df, bcd \to ef, f \to c\}$. Then, $\texttt{Cls}(af, \Sigma) = (acdf, \{b \to e\})$ where $\phi(af) = acdf$ and $\Sigma' = \{b \to e\}$ has important information that will be used in the computation of minimal generators.

For $X, Y \subseteq M$ satisfying that $X = \phi(Y)$, it is usual to say that Y is a generator of the closed set X. Notice that any subset of X containing Y is also a generator of X. When the base set M of a closure system is finite, the set of generators of a closed set can be characterized by its minimal ones.

Definition 7 (Minimal Generator). *Let $\langle M, \phi \rangle$ be a closure system. $X \subseteq M$ is said to be a minimal generator if, for all proper subsets $Y \subset X$, one has $\phi(Y) \subset \phi(X)$.*

In [5], a method to compute all minimal generators is presented. This method named `MinGen` is based on above mentioned closure algorithm `Cls` and it takes the advantages of the additional information that it provides. That is, `Cls` is used not only to compute closed sets from generators but also a smaller implicational set which guides us in the search of new subsets to be considered as minimal generators. The input of `MinGen` is a subset of M and a set of implications Σ and the output is the set of closed sets endowed with all the minimal generators that generate them, i.e. $\{\langle C, mg(C) \rangle : C \text{ is a closed set}\}$ where $mg(C)$ is the set of minimal generators D satisfying $\phi(D) = C$. It can be seen as the lattice of the closure system in which we have added labels with the minimal generators to each closed set.

Example 2. Let $M = \{a, b, c, d\}$ and $\Sigma = \{a \to b, c \to bd, bd \to ac\}$. Then the MinGen algorithm returns the following set:

$$\texttt{MinGen}(M, \Sigma) = \{\langle abcd, \{c, ad, bd\}\rangle, \langle ab, \{a\}\rangle, \langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle \emptyset, \{\emptyset\}\rangle\}$$

Observe that there is trivial information (e.g. $\langle b, \{b\}\rangle$) included in the output of this algorithm. In [5], a version of MinGen, named MinGen_0, is also presented. The difference between this algorithm and the previous one is that here only non-trivial generators are calculated.

For the same input, $\texttt{MinGen}_0(M, \Sigma) = \{\langle abcd, \{c, ad, bd\}\rangle, \langle ab, \{a\}\rangle, \langle \emptyset, \{\emptyset\}\rangle\}$

3 Covers and Generators: Relationships

The definition of a D-basis is strongly based on the notion of a proper minimal cover, and the latter is connected with the notion of a minimal generator. In this paper we are going to work with covers instead of proper covers because our idea is to manage, in a uniform way, the binary implications and non-binary ones. Throughout this section we will assume that M is the base set of a closure system defined either by closure operator ϕ, or by means of a set of implications Σ and its related operator ϕ_Σ. The following definition introduces a notion that extends the concept of a proper cover.

Definition 8 (Cover). *Given $X \subseteq M$ and $x \in M$, we say X is a cover of x, denoted $x \sim X$, if $x \in \phi(X) \setminus X$.*

The following proposition, whose proof is straightforward from definitions, illustrates the relationship among proper covers, covers and minimal generators.

Proposition 2. *Let Σ be a set of implications. For each set $C \subseteq M$ closed with respect to ϕ_Σ and each $X \subsetneq C$,*

1. *X is a generator of C if and only if X is a cover of any $x \in C \setminus X$.*
2. *If X is a proper cover of $x \in M$ then X is also a cover of x.*

Observe that the converse of the second item is not true, e.g. ad is a cover of b ($b \sim ad$) in Example 2 but it is not a proper cover. As we have remarked in Section 2, the above definition of a cover does not coincide with the one introduced in [1].

Corollary 1. *Let Σ be a set of implications. For each set $C \subseteq M$ closed with respect to ϕ_Σ and each $X \subsetneq C$, if X is a minimal generator of C then X is a cover of any $x \in C \setminus X$.*

The following example illustrates these relationships and gives a counterexample to the converse statement in the previous corollary.

Example 3. Let $M = \{a, b, c, d, e\}$ and $\Sigma = \{a \to d, bce \to ad, bde \to ac, ade \to bc, cd \to abe, abd \to ce\}$. In Table 1, the covers of each element of M and the non-trivial minimal generators of each ϕ_Σ-closed set are shown. Moreover, underlined covers are not proper covers whereas the others are.

Attribute	Covers
a	$cd, bcd, bce, bde, cde, bcde$
b	$ac, ae, cd, acd, ace, ade, cde, acde$
c	$ab, ae, abd, abe, ade, bde, abde$
d	$\underline{a}, \underline{ab}, \underline{ac}, \underline{ae}, \underline{abc}, \underline{abe}, \underline{ace}, bce, \underline{abce}$
e	$ab, ac, cd, abc, abd, acd, bcd, abcd$

Closed Sets	Minimal Generators
$abcde$	ab, ac, ae, bce, bde, cd
ad	a

Table 1. Covers, Proper Covers and Minimal Generators.

Observe that, by Proposition 2, X is a cover of an element x if and only if there exist a closed set C and a minimal generator Y of C such that $Y \subseteq X$ and $x \in C \smallsetminus X$. For example, cd is a minimal generator of $abcde$ and $cd \subseteq cde$, thus, cde is a cover of a but not a minimal generator.

The closure operator ϕ^* allows us to introduce the notion of a minimal cover. It will be used later to avoid redundancies in the implications of the basis when we look for an optimal basis.

Definition 9 (Minimal Cover). *Let Σ be a set of implications. Given $X \subseteq M$ and $x \in M$, X is a minimal cover of x if X is a cover and satisfies one of the following conditions:*

1. *X is a singleton, i.e., $\mid X \mid = 1$.*
2. *for every cover Y of x, $Y \subseteq \phi^*(X)$ implies $X \subseteq Y$.*

From this definition we directly obtain the following proposition that relates minimal generators and minimal covers.

Proposition 3. *Let Σ be a set of implications, $X \subseteq M$ and $x \in M$. If X is a minimal cover of $x \in \phi(X) \setminus X$, then X is a minimal generator of $\phi(X)$.*

The following example illustrates the definition of minimal covers and shows that the converse statement to the above proposition is not true.

Example 4. Given the base set and the set of implications of Example 3, Table 2 shows the minimal covers of each element.

Element	Minimal Covers
a	cd, bce, bde
b	ae, cd
c	ab, ae, bde
d	a, bce
e	ab, cd

Table 2. Minimal Covers.

Although ae is a minimal generator of $abcde$, it is not a minimal cover for all $x \in \phi(\{a, b, c, d, e\}) \setminus \{a, e\} = \{b, c, d\}$ but only for b and c.

As the previous examples show, a minimal cover is always a minimal generator, but not conversely. Thus, if we compute all minimal generators for some $x \in M$, we have, at the same time, all minimal covers for x, and the latter are exactly what we need for the computation of the D-basis.

4 *D*-basis by means of Minimal Generators

We remark that it is an open problem to design an algorithm rendering the D-basis from an arbitrary implicational system. In this section, we are going to introduce such an algorithm based on the strong connection between minimal covers and minimal generators. This is shown in Algorithm 1. To begin with, we define the Function `MinimalCovers` to make the algorithm easier to understand.

The input of the Function `MinimalCovers` is a set of covers of a given $x \in M$ and it returns a subset with all its minimal covers.

Function MinimalCovers(L)

input : A set of covers L
output: The set of minimal covers
begin
  foreach $g \in L$ **do**
    foreach $h \in L \setminus g$ **do**
      if $h \subseteq g$ **then**
        remove g from L
      else if $\mid g \mid \neq 1$ **and** $h \subseteq \bigcup_{x \in g} \phi(x)$ **then**
        remove g from L
  return L

Notice that although the condition $h \subseteq g$ implies $h \subseteq \bigcup_{x \in g} \phi(x)$ and both of them lead to the same action, it is more efficient to split them off because the cost of the first one is lower. Thus, we previously check the first one and, if it is not fulfilled, then the other one is tested.

As we have mentioned before, our goal is to obtain the D-basis from an arbitrary implicational system. In the first stage, the algorithm computes all minimal generators by means of the MinGen_0 method proposed in [5]. Then, in a second stage, it associates each minimal generator with all elements for which it is a cover. In this stage, we have calculated a subset of covers containing all minimal covers of a given attribute a. Finally, the algorithm removes those intermediate covers which are not minimal covers, obtaining the D-basis. This sequence of tasks is illustrated in Figure 1.

As Figure 1 depicts, the transition from stage 1 to stage 2 needs a way to associate the minimal generators with some of the elements in its closed set.

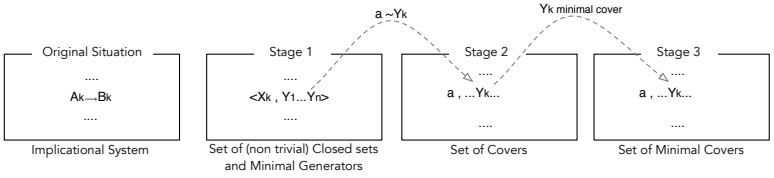


Fig. 1. Stages of D-basis algorithm

Thus, let a be an attribute and mg be the set of minimal generators such that its closure contains a. We write this association as a pair $\langle a, mg\rangle$. Let Φ be a set of such pairs of attributes with their generators. We define the Function `Add` which builds the set of covers produced in Stage 2 as follows:

$$\texttt{Add}(\langle a, mg\rangle, \Phi) = \{\langle a, \{g \in mg | a \notin g\} \cup \{mg_a\}\rangle : \langle a, mg_a\rangle \in \Phi\}$$

Then, in stage 3, the algorithm picks up the set of minimal covers from the set obtained in stage 2 using the Function `MinimalCovers`. The method ends with the Function `OrderedComp` which applies Composition Rule at the same time that it orders the implications in the following sense: the first implications in the D-basis are the binary ones (those with the left-hand side being a singleton).

Algorithm 1: D-basis

input : An implicational system Σ on M
output: The D-basis Σ_D on M
begin
  MinGen:=$\texttt{MinGen}_0(M, \Sigma)$
  C:= $\emptyset$
  foreach $\langle C, mg(C)\rangle \in MinGen$ **do**
    foreach $a \in C$ **do**
      C:=$\texttt{Add}(\langle a, mg(C)\rangle$, C)
  $\Sigma_D := \emptyset$
  foreach $\langle a, mg_a\rangle \in C$ **do**
    mg_a :=`MinimalCovers`(mg_a)
    foreach $g \in mg_a$ **do** $\Sigma_D := \Sigma_D \cup \{g \to a\}$;
  OrderedComp(Σ_D)
  return Σ_D

Example 5. Algorithm 1 returns the following D-basis from the input implicational system of Example 3:

$$\Sigma_D = \{a \to d, bce \to ad, ab \to ce, ae \to bc, bde \to ac, cd \to abe\}$$

We emphasize that although ac is a minimal generator, it is not a minimal cover, thus an implication with ac in the left-hand side is redundant (deduced from inference axioms) and hence should not appear in the D-basis.

A detailed illustrative example

In the conclusion of this section we show the execution of the method, in all its stages, on a set of implications from [3], which was used later to illustrate the D-basis definition in [1].

$$\Sigma = \{5 \to 4, 23 \to 4, 24 \to 3, 34 \to 2, 14 \to 235, 25 \to 134, 35 \to 124, 15 \to 24, 123 \to 45\}$$

As a first step in the algorithm, $\texttt{MinGen}_0$ renders the following set of pairs of closed sets and its non-trivial minimal generators, see Figure 2:

$$\{\langle 12345, \{123, 14, 15, 25, 35\}\rangle, \langle 234, \{23, 24, 34\}\rangle, \langle 45, \{5\}\rangle, \langle \emptyset, \emptyset\rangle\}$$

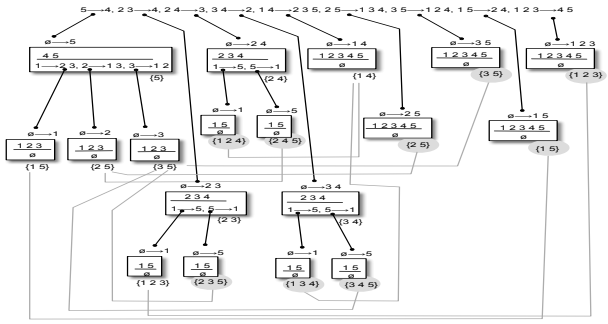


Fig. 2. $\texttt{MinGen}_0$ Execution

Then, for each closed set and each of its elements, our algorithm renders the following set of pairs of elements and covers:

$$\{\langle 1, \{25, 35\}\rangle, \langle 2, \{14, 15, 35, 34\}\rangle, \langle 3, \{14, 15, 25, 24\}\rangle,$$
$$\langle 4, \{123, 15, 25, 35, 5, 23\}\rangle, \langle 5, \{123, 14\}\rangle\}$$

For each element, the Function `MinimalCovers` picks up its minimal covers:

$$\{\langle 1, \{25, 35\}\rangle, \langle 2, \{14, 34\}\rangle, \langle 3, \{14, 24\}\rangle, \langle 4, \{5, 23\}\rangle, \langle 5, \{14, 123\}\rangle\}$$

Finally, at the last step, the algorithm turns these pairs into implications and applies ordered composition resulting in the D-basis.

$$\Sigma_D = \{5 \to 4, 23 \to 4, 24 \to 3, 34 \to 2, 14 \to 235, 25 \to 1, 35 \to 1, 123 \to 5\}$$

5 Conclusion and future works

In this work we have presented a way to obtain the D-basis from any implicational system. In [1] the algorithm was proposed to compute the D-basis from any direct basis, but the computation from any implicational system was left open. There exists also an efficient algorithm for the computation of the D-basis from the context using the method of finding the minimal transversals of the associated hypergraphs [2], but this assumes the different input for the closure system which is outside the scope of this paper.

The Function `MinimalCovers` renders the D-basis within the framework of the closure systems without the need of any transformation. A key point of our work is the connection between covers and generators. Using minimal generators, the D-basis is obtained by reducing the set of minimal generators and transforming it into a set of minimal covers.

As future work, we propose to develop an algorithm which computes the D-basis with better integration of the minimal generator computation to render the minimal covers in a more direct way. In addition, we are planning to design an empirical study and to make a comparison between this algorithm and other techniques proposed in previous papers.

Acknowledgment

Supported by Grants TIN2011-28084 and TIN2014-59471-P of the Science and Innovation Ministry of Spain.

References

1. K. Adaricheva and J. B. Nation and R. Rand, *Ordered direct implicational basis of a finite closure system*, Discrete Applied Mathematics, 161 (6): 707–723, 2013.
2. K. Adaricheva and J.B. Nation, *Discovery of the D-basis in binary tables based on hypergraph dualization*, `http://arxiv.org/abs/1504.02875`, 2015.
3. K. Bertet, B. Monjardet, *The multiple facets of the canonical direct unit implicational basis*, Theor. Comput. Sci., 411(22-24): 2155–2166, 2010.
4. P. Cordero, A Mora, M. Enciso, I.Pérez de Guzmán, *SLFD Logic: Elimination of Data Redundancy in Knowledge Representation*, LNCS, 2527: 141–150, 2002.
5. P. Cordero, M. Enciso, A Mora, M. Ojeda-Aciego, *Computing Minimal Generators from Implications: a Logic-guided Approach*, CLA 2012: 187–198, 2012.
6. V. Duquenne, Some variations on Alan Day's Algorithm for Calculating Canonical Basis of Implications, CLA 2007: 192–207, 2007.
7. B. Ganter, *Two basic algorithms in concept analysis*, Technische Hochschule, Darmstadt, 1984.
8. A. Mora, M. Enciso, P. Cordero, I. Fortes, *Closure via functional dependence simplification*, International Journal of Computer Mathematics, 89(4): 510–526, 2012.
9. S. Rudolph, *Some Notes on Managing Closure Operators*, LNCS, 7278: 278–291, 2012.
10. M. Wild, The joy of implications, aka pure Horn functions: mainly a survey, `http://arxiv.org/abs/1411.6432`, 2014.

Using Linguistic Hedges in L-rough Concept Analysis

Eduard Bartl and Jan Konecny

Data Analysis and Modeling Lab
Dept. Computer Science, Palacky University, Olomouc
17. listopadu 12, CZ-77146 Olomouc
Czech Republic

Abstract. We enrich concept-forming operators in **L**-rough Concept Analysis with linguistic hedges which model semantics of logical connectives 'very' and 'slightly'. Using hedges as parameters for the concept-forming operators we are allowed to modify our uncertainty when forming concepts. As a consequence, by selection of these hedges we can control the size of concept lattice.

Keywords: Formal concept analysis; concept lattice; fuzzy set; linguistic hedge; rough set; uncertainty.

1 Introduction

In [2] we presented a framework which allows us to work with positive and negative attributes in the fuzzy setting by applying two unipolar scales for intents – a positive one and a negative one. The positive scale is implicitly modeled by an antitone Galois connection while the negative scale is modeled by an isotone Galois connection. In this paper we extend this approach in two ways.

First, we work with uncertain information. To do this we extend formal fuzzy contexts to contain two truth-degrees for each object-attribute pair. The two truth-degrees represent necessity and possibility of the fact that an object has an attribute. The interval between these degrees represents the uncertainty presented in a given data.

Second, we parametrize the concept-forming operators used in the framework by unary operators called truth-stressing and truth-depressing linguistic hedges. Their intended use is to model semantics of statements *'it is very sure that this attribute belongs to a fuzzy set (intent)'* and *'it is slightly possible that an attribute belongs a fuzzy set (intent)'*, respectively. In the paper, we demonstrate how the hedges influence the size of concept lattice.

2 Preliminaries

In this section we summarize the basic notions used in the paper.

Residuated Lattices and Fuzzy Sets

We use complete residuated lattices as basic structures of truth-degrees. A complete residuated lattice [4, 12, 17] is a structure $\mathbf{L} = \langle L, \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ such that $\langle L, \wedge, \vee, 0, 1\rangle$ is a complete lattice, i.e. a partially ordered set in which arbitrary infima and suprema exist; $\langle L, \otimes, 1\rangle$ is a commutative monoid, i.e. $\otimes$ is a binary operation which is commutative, associative, and $a \otimes 1 = a$ for each $a \in L$; $\otimes$ and $\rightarrow$ satisfy adjointness, i.e. $a \otimes b \leqslant c$ iff $a \leqslant b \rightarrow c$. 0 and 1 denote the least and greatest elements. The partial order of $\mathbf{L}$ is denoted by $\leqslant$. Throughout this work, $\mathbf{L}$ denotes an arbitrary complete residuated lattice.

Elements of L are called truth degrees. Operations $\otimes$ (multiplication) and $\rightarrow$ (residuum) play the role of (truth functions of) "fuzzy conjunction" and "fuzzy implication". Furthermore, we define the complement of $a \in L$ as $\neg a = a \rightarrow 0$.

An $\mathbf{L}$-set (or fuzzy set) A in a universe set X is a mapping assigning to each $x \in X$ some truth degree $A(x) \in L$. The set of all $\mathbf{L}$-sets in a universe X is denoted $\mathbf{L}^X$.

The operations with $\mathbf{L}$-sets are defined componentwise. For instance, the intersection of $\mathbf{L}$-sets $A, B \in \mathbf{L}^X$ is an $\mathbf{L}$-set $A \cap B$ in X such that $(A \cap B)(x) = A(x) \wedge B(x)$ for each $x \in X$. An $\mathbf{L}$-set $A \in \mathbf{L}^X$ is also denoted $\{{}^{A(x)}\!/\!x \mid x \in X\}$. If for all $y \in X$ distinct from $x_1, \ldots, x_n$ we have $A(y) = 0$, we also write $\{{}^{A(x_1)}\!/\!x_1, \ldots, {}^{A(x_n)}\!/\!x_n\}$.

An $\mathbf{L}$-set $A \in \mathbf{L}^X$ is called normal if there is $x \in X$ such that $A(x) = 1$. An $\mathbf{L}$-set $A \in \mathbf{L}^X$ is called crisp if $A(x) \in \{0, 1\}$ for each $x \in X$. Crisp $\mathbf{L}$-sets can be identified with ordinary sets. For a crisp A, we also write $x \in A$ for $A(x) = 1$ and $x \notin A$ for $A(x) = 0$.

For $A, B \in \mathbf{L}^X$ we define the degree of inclusion of A in B by $S(A, B) = \bigwedge_{x \in X} A(x) \rightarrow B(x)$. Graded inclusion generalizes the classical inclusion relation. Described verbally, $S(A, B)$ represents a degree to which A is a subset of B. In particular, we write $A \subseteq B$ iff $S(A, B) = 1$. As a consequence, we have $A \subseteq B$ iff $A(x) \leqslant B(x)$ for each $x \in X$.

By $\mathbf{L}^{-1}$ we denote L with dual lattice order. An $\mathbf{L}$-*rough set* A in a universe X is a pair of $\mathbf{L}$-sets $A = \langle \underline{A}, \overline{A}\rangle \in (\mathbf{L} \times \mathbf{L}^{-1})^U$. The $\underline{A}$ is called an *lower approximation* of A and the $\overline{A}$ is called a *upper approximation* of A.[1]

The operations with $\mathbf{L}$-rough sets are again defined componentwise, i.e.

$$\bigcap_{i \in I} \langle \underline{A}, \overline{A}\rangle = \langle \bigcap_{i \in I} \underline{A}, \bigcap_{i \in I}{}^{-1} \overline{A}\rangle = \langle \bigcap_{i \in I} \underline{A}, \bigcup_{i \in I} \overline{A}\rangle,$$
$$\bigcup_{i \in I} \langle \underline{A}, \overline{A}\rangle = \langle \bigcup_{i \in I} \underline{A}, \bigcup_{i \in I}{}^{-1} \overline{A}\rangle = \langle \bigcup_{i \in I} \underline{A}, \bigcap_{i \in I} \overline{A}\rangle.$$

Similarly, the graded subsethood is then applied componentwise

$$S(\langle \underline{A}, \overline{A}\rangle, \langle \underline{B}, \overline{B}\rangle) = S(\underline{A}, \underline{B}) \wedge S^{-1}(\overline{A}, \overline{B}) = S(\underline{A}, \underline{B}) \wedge S(\overline{B}, \overline{A})$$

[1] In our setting we consider intents to be $\mathbf{L}$-rough sets; the lower and upper approximation are interpreted as necessary intent and possible intent, respectively.

and the crisp subsethood is then defined using the graded subsethood:

$$\langle \underline{A}, \overline{A} \rangle \subseteq \langle \underline{B}, \overline{B} \rangle \text{ iff } S(\langle \underline{A}, \overline{A} \rangle, \langle \underline{B}, \overline{B} \rangle) = 1, \text{ iff } \underline{A} \subseteq \underline{B} \text{ and } \overline{B} \subseteq \overline{A}.$$

An **L**-rough set $\langle \underline{A}, \overline{A} \rangle$ is called *natural* if $\underline{A} \subseteq \overline{A}$.

Binary **L**-relations (binary fuzzy relations) between X and Y can be thought of as **L**-sets in the universe $X \times Y$. That is, a binary **L**-relation $I \in \mathbf{L}^{X \times Y}$ between a set X and a set Y is a mapping assigning to each $x \in X$ and each $y \in Y$ a truth degree $I(x, y) \in L$ (a degree to which x and y are related by I). **L**-rough relations are then $(\mathbf{L} \times \mathbf{L}^{-1})$-sets in $X \times Y$. For **L**-relation $I \in \mathbf{L}^{X \times Y}$ we define its inverse $I^{-1} \in \mathbf{L}^{Y \times X}$ as $I^{-1}(y, x) = I(x, y)$ for all $x \in X$, $y \in Y$.

Formal Concept Analysis in the Fuzzy Setting

An **L**-context is a triplet $\langle X, Y, I \rangle$ where X and Y are (ordinary) sets and $I \in \mathbf{L}^{X \times Y}$ is an **L**-relation between X and Y. Elements of X are called objects, elements of Y are called attributes, I is called an incidence relation. $I(x, y) = a$ is read: "The object x has the attribute y to degree a."

Consider the following pairs of operators induced by an **L**-context $\langle X, Y, I \rangle$. First, the pair $\langle \uparrow, \downarrow \rangle$ of operators $^{\uparrow} : \mathbf{L}^X \to \mathbf{L}^Y$ and $^{\downarrow} : \mathbf{L}^Y \to \mathbf{L}^X$ is defined by

$$A^{\uparrow}(y) = \bigwedge_{x \in X} A(x) \to I(x, y) \quad \text{and} \quad B^{\downarrow}(x) = \bigwedge_{y \in Y} B(y) \to I(x, y).$$

Second, the pair $\langle \cap, \cup \rangle$ of operators $^{\cap} : \mathbf{L}^X \to \mathbf{L}^Y$ and $^{\cup} : \mathbf{L}^Y \to \mathbf{L}^X$ is defined by

$$A^{\cap}(y) = \bigvee_{x \in X} A(x) \otimes I(x, y) \quad \text{and} \quad B^{\cup}(x) = \bigwedge_{y \in Y} I(x, y) \to B(y).$$

To emphasize that the operators are induced by I, we also denote the operators by $\langle \uparrow_I, \downarrow_I \rangle$ and $\langle \cap_I, \cup_I \rangle$.

Fixpoints of these operators are called formal concepts. The set of all formal concepts (along with set inclusion) forms a complete lattice, called **L**-concept lattice. We denote the sets of all concepts (as well as the corresponding **L**-concept lattice) by $\mathcal{B}^{\uparrow\downarrow}(X, Y, I)$ and $\mathcal{B}^{\cap\cup}(X, Y, I)$, i.e.

$$\mathcal{B}^{\uparrow\downarrow}(X, Y, I) = \{\langle A, B \rangle \in \mathbf{L}^X \times \mathbf{L}^Y \mid A^{\uparrow} = B,\ B^{\downarrow} = A\},$$
$$\mathcal{B}^{\cap\cup}(X, Y, I) = \{\langle A, B \rangle \in \mathbf{L}^X \times \mathbf{L}^Y \mid A^{\cap} = B,\ B^{\cup} = A\}.$$

For an **L**-concept lattice $\mathcal{B}(X, Y, I)$, where $\mathcal{B}$ is either $\mathcal{B}^{\uparrow\downarrow}$ or $\mathcal{B}^{\cap\cup}$, denote the corresponding sets of extents and intents by $\mathrm{Ext}(X, Y, I)$ and $\mathrm{Int}(X, Y, I)$. That is,

$$\mathrm{Ext}(X, Y, I) = \{A \in \mathbf{L}^X \mid \langle A, B \rangle \in \mathcal{B}(X, Y, I) \text{ for some } B\},$$
$$\mathrm{Int}(X, Y, I) = \{B \in \mathbf{L}^Y \mid \langle A, B \rangle \in \mathcal{B}(X, Y, I) \text{ for some } A\}.$$

An $(\mathbf{L}_1, \mathbf{L}_2)$-Galois connection between the sets X and Y is a pair $\langle f, g \rangle$ of mappings $f : \mathbf{L}_1^X \to \mathbf{L}_2^Y$, $g : \mathbf{L}_2^Y \to \mathbf{L}_1^X$, satisfying

$$S(A, g(B)) = S(B, f(A))$$

for every $A \in \mathbf{L}_1^X, B \in \mathbf{L}_2^Y$.

One can easily observe that the couple $\langle \uparrow, \downarrow \rangle$ forms an $(\mathbf{L}, \mathbf{L})$-Galois connection between X and Y, while $\langle \cap, \cup \rangle$ forms an $(\mathbf{L}, \mathbf{L}^{-1})$-Galois connection between X and Y.

L-*rough Contexts and* **L**-*rough Concepts Lattices*

An **L**-rough context is a quadruple $\langle X, Y, \underline{I}, \overline{I} \rangle$, where X and Y are (crisp) sets of objects and attributes, respectively, and the $\langle \underline{I}, \overline{I} \rangle$ is a **L**-rough relation. The meaning of $\langle \underline{I}, \overline{I} \rangle$ is as follows: $\underline{I}(x, y)$ (resp. $\overline{I}(x, y)$) is the truth degree to which the object x surely (resp. possibly) has the attribute y. The quadruple $\langle X, Y, \underline{I}, \overline{I} \rangle$ is called a **L**-*rough context*.

The **L**-rough context induces two operators defined as follows. Let $\langle X, Y, \underline{I}, \overline{I} \rangle$ be an **L**-rough context. Define **L**-*rough concept-forming operators* as

$$\begin{aligned} A^{\Delta} &= \langle A^{\uparrow_{\underline{I}}}, A^{\cap_{\overline{I}}} \rangle, \\ \langle \underline{B}, \overline{B} \rangle^{\nabla} &= \underline{B}^{\downarrow_{\underline{I}}} \cap \overline{B}^{\cup_{\overline{I}}} \end{aligned} \tag{1}$$

for $A \in \mathbf{L}^X, \underline{B}, \overline{B} \in \mathbf{L}^Y$. Fixed points of $\langle \Delta, \nabla \rangle$, i.e. tuples $\langle A, \langle \underline{B}, \overline{B} \rangle \rangle \in \mathbf{L}^X \times (\mathbf{L} \times \mathbf{L}^{-1})^Y$ such that $A^{\Delta} = \langle \underline{B}, \overline{B} \rangle$ and $\langle \underline{B}, \overline{B} \rangle^{\nabla} = A$, are called **L**-*rough concepts*. The $\underline{B}$ and $\overline{B}$ are called *lower intent approximation* and *upper intent approximation*, respectively.

In [2] we showed that the pair of operators (1) is an $(\mathbf{L}, \mathbf{L} \times \mathbf{L}^{-1})$-Galois connection.

Linguistic Hedges

Truth-stressing hedges were studied from the point of fuzzy logic as logical connectives 'very true', see [13]. Our approach is close to that in [13]. A *truth-stressing hedge* is a mapping $^* : L \to L$ satisfying

$$1^* = 1, \quad a^* \leqslant a, \quad a \leqslant b \text{ implies } a^* \leqslant b^*, \quad a^{**} = a^* \tag{2}$$

for each $a, b \in L$. Truth-stressing hedges were used to parametrize antitone **L**-Galois connections e.g. in [3, 5, 9], and also to parameterize isotone **L**-Galois connections in [1].

On every complete residuated lattice **L**, there are two important truth-stressing hedges:

(i) identity, i.e. $a^* = a$ $(a \in L)$;
(ii) globalization, i.e.

$$a^* = \begin{cases} 1, & \text{if } a = 1, \\ 0, & \text{otherwise.} \end{cases}$$

A *truth-depressing hedge* is a mapping $^{\square} : L \to L$ such that following conditions are satisfied

$$0^{\square} = 0, \quad a \leqslant a^{\square}, \quad a \leqslant b \text{ implies } a^{\square} \leqslant b^{\square}, \quad a^{\square\square} = a^{\square}$$

for each $a, b \in L$. A truth-depressing hedge is a (truth function of) logical connective 'slightly true', see [16].

On every complete residuated lattice **L**, there are two important truth-depressing hedges:

(i) identity, i.e. $a^{\Box} = a$ $(a \in L)$;
(ii) antiglobalization, i.e.

$$a^{\Box} = \begin{cases} 0, & \text{if } a = 0, \\ 1, & \text{otherwise .} \end{cases}$$

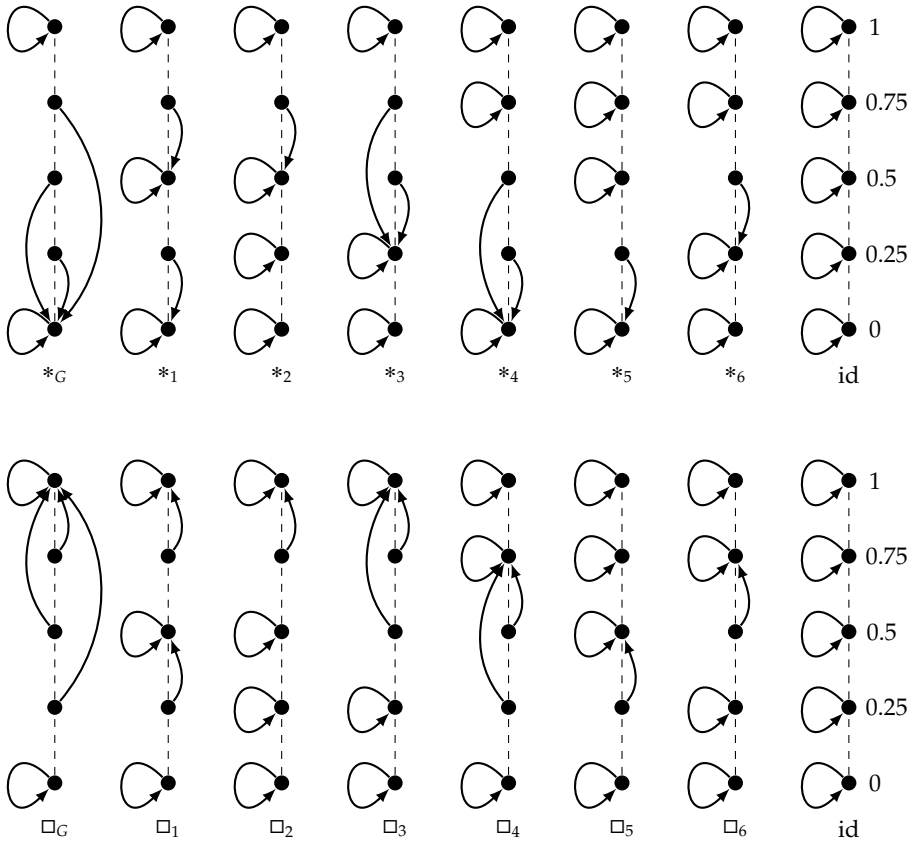


Fig. 1. Truth-stressing hedges (top) and truth-depressing hedges (bottom) on 5-element chain with Łukasiewicz operations $\mathbf{L} = \langle \{0, 0.25, 0.5, 0.75, 1\}, \min, \max, \otimes, \rightarrow, 0, 1 \rangle$. The leftmost truth-stressing hedge $*_G$ is the globalization, leftmost truth-depressing hedge $\Box_G$ is the antiglobalization. The rightmost hedges denoted by id are the identities.

For truth-stressing/truth-depressing hedge $*$ we denote by $\mathrm{fix}(*)$ set of its idempotent elements in **L**; i.e. $\mathrm{fix}(*) = \{a \in L \mid a^* = a\}$.

Let $*_1, *_2$ be truth-stressing hedges on **L** such that $\mathrm{fix}(*_1) \subseteq \mathrm{fix}(*_2)$; then for each $a \in A$, $a^{*_1*_2} = a^{*_1}$ holds. The same holds true for $*_1, *_2$ being truth-depressing hedges.

We naturally extend application of truth-stressing/truth-depressing hedges to **L**-sets: $A^*(x) = A(x)^*$ for all $x \in U$.

3 Results

The **L**-rough concept-forming operator $\triangle$ gives for each **L**-set of objects two **L**-sets of attributes. The first one represents a necessity of having the attributes and second one a possibility of having the attributes. We add linguistic hedges to the concept-forming operators to control shape of the two **L**-sets.

Since the **L**-rough concept-forming operators are defined via $\langle\uparrow, \downarrow\rangle$ and $\langle\cap, \cup\rangle$, we first recall the parametrization of these operators as described in [8, 15].

3.1 Linguistic Hedges in Formal Fuzzy Concept Analysis

Let $\langle X, Y, I\rangle$ be an **L**-context and let $\heartsuit, \diamondsuit$ be truth-stressing hedges on **L**. The antitone concept-forming operators parametrized by $\heartsuit$ and $\diamondsuit$ induced by I are defined as

$$A^{\uparrow_\heartsuit}(y) = \bigwedge_{x\in X} A(x)^\heartsuit \to I(x, y),$$
$$B^{\downarrow_\diamondsuit}(x) = \bigwedge_{y\in Y} B(y)^\diamondsuit \to I(x, y)$$

for all $A \in \mathbf{L}^X, B \in \mathbf{L}^Y$.

Let $\heartsuit$ and $\spadesuit$ be truth-stressing hedge and truth-depressing hedge on **L**, respectively. The isotone concept-forming operators parametrized by $\heartsuit$ and $\spadesuit$ induced by I are defined as

$$A^{\cap_\heartsuit}(y) = \bigvee_{x\in X} A(x)^\heartsuit \otimes I(x, y),$$
$$B^{\cup_\spadesuit}(x) = \bigwedge_{y\in Y} I(x, y) \to B(y)^\spadesuit$$

for all $A \in \mathbf{L}^X, B \in \mathbf{L}^Y$.

Properties of the hedges in the setting of multi-adjoint concept lattices with heterogeneous conjunctors were studied in [14].

3.2 L-rough Concept-Forming Operators with Linguistic Hedges

Let $\heartsuit, \diamondsuit$ be truth-stressing hedges on **L** and let $\spadesuit$ be a truth-depressing hedge on **L**. We parametrize the **L**-rough concept-forming operators as

$$A^\blacktriangle = \langle A^{\uparrow_\heartsuit}, A^{\cap_\heartsuit}\rangle \quad \text{and} \quad \langle \underline{B}, \overline{B}\rangle^\blacktriangledown = \underline{B}^{\downarrow_\diamondsuit} \cap \overline{B}^{\cup_\spadesuit} \tag{3}$$

for $A \in \mathbf{L}^X, \underline{B}, \overline{B} \in \mathbf{L}^Y$.

Remark 1. When the all three hedges are identities the pair $\langle \blacktriangle, \blacktriangledown \rangle$ is equivalent to $\langle \vartriangle, \triangledown \rangle$; so it is an $(\mathbf{L}, \mathbf{L} \times \mathbf{L}^{-1})$-Galois connection. For arbitrary hedges this does not hold.

The following theorem describes properties of $\langle \blacktriangle, \blacktriangledown \rangle$.

Theorem 1. *The pair $\langle \blacktriangle, \blacktriangledown \rangle$ of* **L***-rough concept-forming operators parametrized by hedges has the following properties.*

(a) $A^{\blacktriangle} = A^{\blacktriangledown\vartriangle} = A^{\blacktriangledown\blacktriangle}$ *and* $\langle \underline{B}, \overline{B} \rangle^{\blacktriangledown} = \langle \underline{B}^{\blacklozenge}, \overline{B}^{\blacklozenge} \rangle^{\triangledown} = \langle \underline{B}^{\blacklozenge}, \overline{B}^{\blacklozenge} \rangle^{\blacktriangledown}$

(b) $A^{\vartriangle} \subseteq A^{\blacktriangle}$ *and* $\langle \underline{B}, \overline{B} \rangle^{\triangledown} \subseteq \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown}$

(c) $S(A_1^{\blacktriangledown}, A_2^{\blacktriangledown}) \leqslant S(A_2^{\blacktriangle}, A_1^{\blacktriangle})$ *and* $S(\langle \underline{B_1}, \overline{B_1} \rangle, \langle \underline{B_2}, \overline{B_2} \rangle) \leqslant S(\langle \underline{B_2}, \overline{B_2} \rangle^{\blacktriangledown}, \langle \underline{B_1}, \overline{B_1} \rangle^{\blacktriangledown})$

(d) $A^{\blacktriangledown} \subseteq A^{\blacktriangle\blacktriangledown}$ *and* $\langle \underline{B}^{\blacklozenge}, \overline{B}^{\blacklozenge} \rangle \subseteq \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown\blacktriangle}$;

(e) $A_1 \subseteq A_2$ *implies* $A_2^{\blacktriangle} \subseteq A_1^{\blacktriangle}$ *and* $\langle \underline{B_1}, \overline{B_1} \rangle \subseteq \langle \underline{B_2}, \overline{B_2} \rangle$ *implies* $\langle \underline{B_2}, \overline{B_2} \rangle^{\blacktriangledown} \subseteq \langle \underline{B_1}, \overline{B_1} \rangle^{\blacktriangledown}$

(f) $S(A^{\blacktriangledown}, \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown}) = S(\langle \underline{B}^{\blacklozenge}, \overline{B}^{\blacklozenge} \rangle, A^{\blacktriangle})$

(g) $(\bigcup_{i\in I} A_i^{\blacktriangledown})^{\blacktriangle} = \bigcap_{i\in I} A_i^{\blacktriangle}$ *and* $(\langle \bigcup_{i\in I} \underline{B_i}^{\blacklozenge}, \bigcap_{i\in I} \overline{B_i}^{\blacklozenge} \rangle)^{\blacktriangledown} = \bigcap_{i\in I} \langle \underline{B_i}, \overline{B_i} \rangle^{\blacktriangledown}$

(h) $A^{\blacktriangle\blacktriangledown} = A^{\blacktriangle\blacktriangledown\blacktriangle\blacktriangledown}$ *and* $\langle \underline{B}, \overline{B} \rangle^{\blacktriangledown\blacktriangle} = \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown\blacktriangle\blacktriangledown\blacktriangle}$.

Proof. (a) Follows immediately from definition of $\blacktriangle$ and $\blacktriangledown$ and idempotency of hedges.

(b) From (2) we have $A^{\blacktriangledown} \subseteq A$; by properties of Galois connections the inclusion implies $A^{\vartriangle} \subseteq A^{\blacktriangledown\vartriangle}$, which is by (a) equivalent to $A^{\vartriangle} \subseteq A^{\blacktriangle}$. Proof of the second statement in (b) is similar.

(c) Follows from (a) and properties of Galois connections.

(d) By [2, Corollary 1(a)] we have $A^{\blacktriangledown} \subseteq A^{\blacktriangledown\vartriangle\triangledown}$. Using (a) we get $A^{\blacktriangledown} \subseteq A^{\blacktriangle\triangledown}$ and from (b) we have $A^{\blacktriangle\triangledown} \subseteq A^{\blacktriangle\blacktriangledown}$, so $A^{\blacktriangledown} \subseteq A^{\blacktriangle\blacktriangledown}$. Similarly for the second claim.

(e) Follows directly from [2, Corollary 1(c)] and properties of Galois connections.

(f) Since $\langle \vartriangle, \triangledown \rangle$ forms $(\mathbf{L}, \mathbf{L} \times \mathbf{L}^{-1})$-Galois connection and using (a) we have $S(A^{\blacktriangledown}, \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown}) = S(A^{\blacktriangledown}, \langle \underline{B}^{\blacklozenge}, \overline{B}^{\blacklozenge} \rangle^{\triangledown}) = S(\langle \underline{B}^{\blacklozenge}, \overline{B}^{\blacklozenge} \rangle, A^{\blacktriangledown\vartriangle}) = S(\langle \underline{B}^{\blacklozenge}, \overline{B}^{\blacklozenge} \rangle, A^{\blacktriangle})$.

(g) We can easily get

$$\begin{aligned}(\bigcup_{i\in I} A_i^{\blacktriangledown})^{\blacktriangle} &= \langle (\bigcup_{i\in I} A_i^{\blacktriangledown})^{\uparrow\blacktriangledown}, (\bigcup_{i\in I} A_i^{\blacktriangledown})^{\cap\blacktriangledown} \rangle = \langle \bigcap_{i\in I} A_i^{\uparrow\blacktriangledown}, \bigcup_{i\in I} A_i^{\cap\blacktriangledown} \rangle \\ &= \bigcap_{i\in I} \langle A_i^{\uparrow\blacktriangledown}, A_i^{\cap\blacktriangledown} \rangle = \bigcap_{i\in I} A_i^{\blacktriangle},\end{aligned}$$

and

$$\begin{aligned}(\langle \bigcup_{i\in I} \underline{B_i}^{\blacklozenge}, \bigcap_{i\in I} \overline{B_i}^{\blacklozenge} \rangle)^{\blacktriangledown} &= (\bigcup_{i\in I} \underline{B_i}^{\blacklozenge})^{\downarrow\blacklozenge} \cap (\bigcap_{i\in I} \overline{B_i}^{\blacklozenge})^{\cup\blacktriangle} = \bigcap_{i\in I} \underline{B_i}^{\downarrow\blacklozenge} \cap \bigcap_{i\in I} \overline{B_i}^{\cup\blacktriangle} \\ &= \bigcap_{i\in I} (\underline{B_i}^{\downarrow\blacklozenge} \cap \overline{B_i}^{\cup\blacktriangle}) = \bigcap_{i\in I} \langle \underline{B_i}, \overline{B_i} \rangle^{\blacktriangledown}.\end{aligned}$$

(h) Using (a), (d) and (e) twice, we have $A^{\blacktriangle\blacktriangledown} \subseteq A^{\blacktriangle\blacktriangledown\blacktriangle\blacktriangledown}$. Using (d) for $\langle \underline{B}, \overline{B} \rangle = A^{\blacktriangle}$ we have $A^{\blacktriangle\blacktriangledown} \subseteq A^{\blacktriangle\blacktriangledown\blacktriangle} = A^{\blacktriangle\blacktriangledown\blacktriangle}$. Then applying (e) we get $A^{\blacktriangle\blacktriangledown\blacktriangle\blacktriangledown} \subseteq A^{\blacktriangle\blacktriangledown}$ proving the first claim. The second claim can be proved analogically. □

The set of fixed points of $\langle \blacktriangle, \blacktriangledown \rangle$ endowed with partial order $\leqslant$ given by

$$\begin{aligned} \langle A_1, \underline{B}_1, \overline{B}_1 \rangle \leqslant \langle A_2, \underline{B}_2, \overline{B}_2 \rangle \quad &\text{iff} \quad A_1 \subseteq A_2 \\ &\text{iff} \quad \langle \underline{B}_1, \overline{B}_1 \rangle \subseteq \langle \underline{B}_2, \overline{B}_2 \rangle \end{aligned} \tag{4}$$

is denoted by $\mathcal{B}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\spadesuit}(X, Y, \underline{I}, \overline{I})$.

Remark 2. Note that from (4) it is clear that if a concept has non-natural **L**-rough intent then all its subconcepts have non-natural intent. If such concepts are not desired, one can simply ignore them and work with the iceberg lattice of concepts with natural **L**-rough intents.

The next theorem shows a crisp representation of $\mathcal{B}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\spadesuit}(X, Y, \underline{I}, \overline{I})$.

Theorem 2. *$\mathcal{B}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\spadesuit}(X, Y, \underline{I}, \overline{I})$ is isomorphic to ordinary concept lattice $\mathcal{B}^{\uparrow\downarrow}(X \times \mathrm{fix}(\heartsuit), Y \times \mathrm{fix}(\diamondsuit) \times \mathrm{fix}(\spadesuit), I^{\times})$ where*

$$\langle\langle x, a\rangle, \langle y, \underline{b}, \overline{b}\rangle\rangle \in I^{\times} \text{ iff } a \otimes \underline{b} \leqslant \underline{I}(x, y) \text{ and } a \to \overline{b} \geqslant \overline{I}(x, y).$$

Proof. This proof can be done by following the same steps as in [8, 15]. □

The following theorem explains the structure of $\mathcal{B}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\spadesuit}(X, Y, \underline{I}, \overline{I})$.

Theorem 3. *$\mathcal{B}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\spadesuit}(X, Y, \underline{I}, \overline{I})$ is a complete lattice with suprema and infima defined as*

$$\begin{aligned} \bigwedge_i \langle A_i, \langle \underline{B_i}, \overline{B_i} \rangle \rangle &= \langle (\bigcap_i A_i)^{\blacktriangle\blacktriangledown}, \langle \bigcup_i \underline{B_i}^{\diamondsuit}, \bigcap_i \overline{B_i}^{\spadesuit} \rangle^{\blacktriangledown\blacktriangle} \rangle, \\ \bigvee_i \langle A_i, \langle \underline{B_i}, \overline{B_i} \rangle \rangle &= \langle (\bigcup_i A_i^{\heartsuit})^{\blacktriangle\blacktriangledown}, \langle \bigcap_i \underline{B_i}, \bigcup_i \overline{B_i} \rangle^{\blacktriangledown\blacktriangle} \rangle \end{aligned}$$

for all $A_i \in \mathbf{L}^X, \underline{B_i} \in \mathbf{L}^Y, \overline{B_i} \in \mathbf{L}^Y$.

Proof. Follows from Theorem 2. □

Remark 3. Note that if we alternatively define (3) as

$$A^{\blacktriangle} = \langle (A^{\uparrow\blacktriangledown})^{\blacklozenge}, (A^{\cap\blacktriangledown})^{\blacklozenge} \rangle \quad \text{and} \quad \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown} = (\underline{B}^{\downarrow\blacklozenge} \cap \overline{B}^{\cup\blacklozenge})^{\heartsuit} \tag{5}$$

or

$$A^{\blacktriangle} = \langle (A^{\uparrow})^{\blacklozenge}, (A^{\cap})^{\blacklozenge} \rangle \quad \text{and} \quad \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown} = (\underline{B}^{\downarrow} \cap \overline{B}^{\cup})^{\heartsuit} \tag{6}$$

or

$$A^{\blacktriangle} = \langle (A^{\uparrow\blacktriangledown})^{\blacklozenge}, (A^{\cap\blacktriangledown})^{\blacklozenge} \rangle \quad \text{and} \quad \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown} = \langle \underline{B}, \overline{B} \rangle^{\triangledown}$$

or

$$A^{\blacktriangle} = A^{\triangle} \quad \text{and} \quad \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown} = (\underline{B}^{\downarrow\blacklozenge} \cap \overline{B}^{\cup\blacklozenge})^{\heartsuit}$$

we obtain an isomorphic concept lattice. In addition (5) and (6) produce the same concept lattice.

3.3 Size Reduction of Fuzzy Rough Concept Lattices

This part provides analogous results on reduction with truth-stressing and truth-depressing hedges as [10] for antitone fuzzy concept-forming operators and [15] for isotone fuzzy concept-forming operators.

For the next theorem we need the following lemma.

Lemma 1. *Let* $\blacktriangle_\heartsuit, \blacktriangle_\diamondsuit, \blacktriangle_\spadesuit, \blacktriangle_\clubsuit$ *be truth-stressing hedges on* $\mathbf{L}$ *such that* $\mathrm{fix}(\blacktriangle_\heartsuit) \subseteq \mathrm{fix}(\blacktriangle_\diamondsuit), \mathrm{fix}(\blacktriangle_\spadesuit) \subseteq \mathrm{fix}(\blacktriangle_\clubsuit)$*; let* $\blacktriangledown_\diamondsuit, \blacktriangledown_\clubsuit$ *be truth-depressing hedges on* $\mathbf{L}$ *such that and* $\mathrm{fix}(\blacktriangledown_\diamondsuit) \subseteq \mathrm{fix}(\blacktriangledown_\clubsuit)$*. We have*

$$A^{\blacktriangle_\heartsuit} \subseteq A^{\blacktriangle_\spadesuit} \quad \text{and} \quad \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown_\diamondsuit, \blacktriangledown_\clubsuit} \subseteq \langle \underline{B}, \overline{B} \rangle^{\blacktriangledown_\spadesuit, \blacktriangledown_\clubsuit}.$$

Proof. We have $A^{\blacktriangle_\spadesuit \blacktriangle_\heartsuit} \subseteq A^{\blacktriangle_\heartsuit}$ from (2). From the assumption $\mathrm{fix}(\blacktriangle_\spadesuit) \subseteq \mathrm{fix}(\blacktriangle_\heartsuit)$ we get $A^{\blacktriangle_\spadesuit \blacktriangle_\heartsuit} = A^{\blacktriangle_\spadesuit}$; whence we have $A^{\blacktriangle_\spadesuit} \subseteq A^{\blacktriangle_\heartsuit}$. Theorem 1(e) implies $A^{\blacktriangle_\heartsuit \blacktriangle_\diamondsuit} \subseteq A^{\blacktriangle_\spadesuit \blacktriangle_\diamondsuit}$ which is by the claim (a) of this theorem equivalent to $A^{\blacktriangle_\diamondsuit} \subseteq A^{\blacktriangle_\spadesuit}$. The second claim can be proved similarly. □

Theorem 4. *Let* $\blacktriangle_\heartsuit, \blacktriangle_\diamondsuit, \blacktriangle_\spadesuit, \blacktriangle_\clubsuit$ *be truth-stressing hedges on* $\mathbf{L}$ *such that* $\mathrm{fix}(\blacktriangle_\heartsuit) \subseteq \mathrm{fix}(\blacktriangle_\diamondsuit)$, $\mathrm{fix}(\blacktriangle_\spadesuit) \subseteq \mathrm{fix}(\blacktriangle_\clubsuit)$*; let* $\blacktriangledown_\diamondsuit, \blacktriangledown_\clubsuit$ *be truth-depressing hedges on* $\mathbf{L}$ *s.t. and* $\mathrm{fix}(\blacktriangledown_\diamondsuit) \subseteq \mathrm{fix}(\blacktriangledown_\clubsuit)$*,*

$$|\mathcal{B}^{\blacktriangle\blacktriangledown}_{\heartsuit,\spadesuit}(X, Y, \underline{I}, \overline{I})| \leqslant |\mathcal{B}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\clubsuit}(X, Y, \underline{I}, \overline{I})|$$

for all $\mathbf{L}$*-rough contexts* $\langle X, Y, \underline{I}, \overline{I} \rangle$*.*

In addition, if $\blacktriangle_\heartsuit = \blacktriangle_\diamondsuit = \mathrm{id}$*, we have*

$$\mathrm{Ext}^{\blacktriangle\blacktriangledown}_{\heartsuit,\spadesuit}(X, Y, \underline{I}, \overline{I}) \subseteq \mathrm{Ext}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\clubsuit}(X, Y, \underline{I}, \overline{I}).$$

Similarly, if $\blacktriangledown_\diamondsuit = \blacktriangledown_\clubsuit = \mathrm{id}$*, we have*

$$\mathrm{Int}^{\blacktriangle\blacktriangledown}_{\heartsuit,\spadesuit}(X, Y, \underline{I}, \overline{I}) \subseteq \mathrm{Int}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\clubsuit}(X, Y, \underline{I}, \overline{I}).$$

Proof. (4) follows directly from Theorem 2 and results on subcontexts in [11].

Now, we show (4). Note that each $A \in \mathrm{Ext}^{\blacktriangle\blacktriangledown}_{\heartsuit,\spadesuit}(X, Y, \underline{I}, \overline{I})$ we have

$$A = A^{\blacktriangle_\spadesuit \blacktriangledown_\clubsuit} = A^{\blacktriangle_\heartsuit \blacktriangledown_\clubsuit} \supseteq A^{\blacktriangle_\heartsuit \blacktriangledown_\diamondsuit} \supseteq A.$$

Thus we have $A \in \mathrm{Ext}^{\blacktriangle\blacktriangledown}_{\heartsuit,\diamondsuit,\clubsuit}(X, Y, \underline{I}, \overline{I})$. The inclusion (4) can be proved similarly. □

Example 1. Consider the truth-stressing hedges $*_G, *_1, *_2$, id and truth-depressing hedges $\square_G, \square_1, \square_2$, id from Figure 1. One can easily observe that

$$\mathrm{fix}(*_G) \subseteq \mathrm{fix}(*_1) \subseteq \mathrm{fix}(*_2) \subseteq \mathrm{fix}(\mathrm{id})$$
$$\mathrm{fix}(\square_G) \subseteq \mathrm{fix}(\square_1) \subseteq \mathrm{fix}(\square_2) \subseteq \mathrm{fix}(\mathrm{id}).$$

Consider the $\mathbf{L}$-context of books and their graded properties in Fig. 2 with $\mathbf{L}$ being 5-element Łukasiewicz chain. Using various combinations of the hedges we obtain a smooth transition in size of the associated fuzzy rough concept lattice going from 10 concepts up to 498 (see Tab. 1). When the 5-element Gödel chain is used instead, we again get a transition going from 10 concepts up to 298 (see Tab. 2).

	High rating	Large no. of pages	Low price	Top sales rank
1	0.75	0	1	0
2	0.5	1	0.25	0.5
3	1	1	0.25	0.5
4	0.75	0.5	0.25	1
5	0.75	0.25	0.75	0
6	1	0	0.75	0.25

Fig. 2. **L**-context of books and their graded properties; this **L**-context was used in [1, 15] to demonstrate reduction of **L**-concept lattices using hedges.

$\spadesuit = \Box_G$	$*_G$	$*_1$	$*_2$	id
$*_G$	10	16	59	61
$*_1$	12	22	65	93
$*_2$	15	26	69	103
id	19	41	97	152

$\spadesuit = \Box_1$	$*_G$	$*_1$	$*_2$	id
$*_G$	15	28	71	110
$*_1$	15	28	71	170
$*_2$	22	28	79	195
id	28	28	110	264

$\spadesuit = \Box_2$	$*_G$	$*_1$	$*_2$	id
$*_G$	15	53	134	211
$*_1$	15	53	134	290
$*_2$	22	63	146	327
id	28	80	181	415

$\spadesuit = \mathrm{id}$	$*_G$	$*_1$	$*_2$	id
$*_G$	27	75	160	297
$*_1$	27	75	160	372
$*_2$	32	80	165	396
id	40	99	202	498

Table 1. Numbers of concepts in **L**-context from Fig. 2 formed by $\langle \blacktriangle, \blacktriangledown \rangle$ parametrized by $\heartsuit, \diamondsuit$, and $\spadesuit$. A 5-element Łukasiewicz chain is used as the structure of truth degrees. The rows represent the hedge $\heartsuit$ and the columns represent the hedge $\diamondsuit$.

$\spadesuit = \Box_G$	$*_G$	$*_1$	$*_2$	id
$*_G$	10	18	24	24
$*_1$	12	21	33	36
$*_2$	15	29	45	48
id	19	33	51	54

$\spadesuit = \Box_G$	$*_G$	$*_1$	$*_2$	id
$*_G$	15	29	36	45
$*_1$	15	32	49	63
$*_2$	22	57	78	106
id	28	66	89	117

$\spadesuit = \Box_G$	$*_G$	$*_1$	$*_2$	id
$*_G$	15	32	48	59
$*_1$	15	32	59	75
$*_2$	22	57	88	118
id	28	66	100	130

$\spadesuit = \Box_G$	$*_G$	$*_1$	$*_2$	id
$*_G$	27	50	66	125
$*_1$	27	50	80	167
$*_2$	32	79	113	257
id	40	90	127	298

Table 2. Numbers of concepts in **L**-context from Fig. 2 formed by $\langle \blacktriangle, \blacktriangledown \rangle$ parametrized by $\heartsuit, \diamondsuit$, and $\spadesuit$. A 5-element Gödel chain is used as the structure of truth degrees. The rows represent the hedge $\heartsuit$ and the columns represent the hedge $\diamondsuit$.

4 Conclusion and further research

We have shown that the **L**-rough concept-forming operators can be parameterized by truth-stressing and truth-depressing hedges similarly as the antitone and isotone fuzzy concept-forming operators.

Our future research includes a study of attribute implications using whose semantics is related to the present setting. That will combine results on fuzzy attribute implications [7] and attribute containment formulas [6].

Acknowledgment

Supported by grant No. 15-17899S, "Decompositions of Matrices with Boolean and Ordinal Data: Theory and Algorithms", of the Czech Science Foundation.

References

1. Eduard Bartl, Radim Belohlavek, Jan Konecny, and Vilem Vychodil. Isotone Galois connections and concept lattices with hedges. In *IEEE IS 2008, Int. IEEE Conference on Intelligent Systems*, pages 15–24–15–28, Varna, Bulgaria, 2008.
2. Eduard Bartl and Jan Konecny. Formal L-concepts with Rough Intents. In *CLA 2014: Proceedings of the 11th International Conference on Concept Lattices and Their Applications*, pages 207–218, 2014.
3. Radim Belohlavek. Reduction and simple proof of characterization of fuzzy concept lattices. *Fundamenta Informaticae*, 46(4):277–285, 2001.
4. Radim Belohlavek. *Fuzzy Relational Systems: Foundations and Principles*. Kluwer Academic Publishers, Norwell, USA, 2002.
5. Radim Belohlavek, Tatana Funioková, and Vilem Vychodil. Fuzzy closure operators with truth stressers. *Logic Journal of the IGPL*, 13(5):503–513, 2005.
6. Radim Belohlavek and Jan Konecny. A logic of attribute containment, 2008.
7. Radim Belohlavek and Vilem Vychodil. A logic of graded attributes. *submitted to Artificial Intelligence*.
8. Radim Belohlavek and Vilem Vychodil. Reducing the size of fuzzy concept lattices by hedges. In *FUZZ-IEEE 2005, The IEEE International Conference on Fuzzy Systems*, pages 663–668, Reno (Nevada, USA), 2005.
9. Radim Belohlavek and Vilem Vychodil. Fuzzy concept lattices constrained by hedges. *JACIII*, 11(6):536–545, 2007.
10. Radim Belohlavek and Vilem Vychodil. Formal concept analysis and linguistic hedges. *Int. J. General Systems*, 41(5):503–532, 2012.
11. Bernard Ganter and Rudolf Wille. *Formal Concept Analysis – Mathematical Foundations*. Springer, 1999.
12. Petr Hájek. *Metamathematics of Fuzzy Logic (Trends in Logic)*. Springer, November 2001.
13. Petr Hájek. On very true. *Fuzzy Sets and Systems*, 124(3):329–333, 2001.
14. Jan Konecny, Jesús Medina and Manuel Ojeda-Aciego Multi-adjoint concept lattices with heterogeneous conjunctors and hedges. *Annals of Mathematics and Artificial Intelligence*, 72(1):73–89, 2011.

15. Jan Konecny. Isotone fuzzy Galois connections with hedges. *Information Sciences*, 181(10):1804–1817, 2011. Special Issue on Information Engineering Applications Based on Lattices.
16. Vilem Vychodil. Truth-depressing hedges and BL-logic. *Fuzzy Sets and Systems*, 157(15):2074–2090, 2006.
17. Morgan Ward and R. P. Dilworth. Residuated lattices. *Transactions of the American Mathematical Society*, 45:335–354, 1939.

Revisiting Pattern Structures for Structured Attribute Sets

Mehwish Alam[1], Aleksey Buzmakov[1], Amedeo Napoli[1], and Alibek Sailanbayev[2⋆]

[1]LORIA (CNRS – Inria NGE – U. de Lorraine), Vandœuvre-lès-Nancy, France
[2]Nazarbayev University, Astana, Kazakhstan
{ mehwish.alam, aleksey.buzmakov, amedeo.napoli, } @loria.fr,
alibek.sailanbayev@nu.edu.kz

Abstract. In this paper, we revisit an original proposition on pattern structures for structured sets of attributes. There are several reasons for carrying out this kind of research work. The original proposition does not give many details on the whole framework, and especially on the possible ways of implementing the similarity operation. There exists an alternative definition without any reference to pattern structures, and we would like to make a parallel between two points of view. Moreover we discuss an efficient implementation of the intersection operation in the corresponding pattern structure. Finally, we discovered that pattern structures for structured attribute sets are very well adapted to the classification and the analysis of RDF data. We terminate the paper by an experimental section where it is shown that the provided implementation of pattern structures for structured attribute sets is quite efficient.

Keywords: Formal Concept Analysis, Pattern Structures, Structured Attribute Sets, Least Common Ancestor, Range Minimum Query.

1 Introduction

In this paper, we want to make precise and develop a section of [1] related to pattern structures and structured sets of attributes. There are several reasons for carrying out this kind of research work. Firstly, the the pattern structures, the similarity operator $\sqcap$ and the associated subsumption operator $\sqsubseteq$ for structured sets of attributes are based on antichains and rather briefly sketched in the original paper. Secondly, there is an alternative and a more "qualitative" point of view on the same subject in [2, 3] without any reference to pattern structures, and we would like to make a parallel between these two points of view. Finally, for classifying RDF triples in the analysis of the content of Linked Open Data (LOD), we discovered that actually pattern structures for structured sets of attributes are very well adapted to solve this problem [4]. Moreover, the

⋆ This work was done during the stay of Alibek Sailanbayev at LORIA, France.

classification of RDF triples provides a very good and practical example for illustrating the use of such a pattern structure and helps to reconcile the two above points of view.

Accordingly, in this paper, we will go back to the two original definitions and show how they are related. For completing the history, it is worth mentioning that antichains, whose intersection is the basis of the similarity operation in the pattern structure for structured attribute sets, our paper, are studied in the book [5]. Moreover, this book cites as an application of antichain intersection an older paper from 1994 [6], written in French, about the decomposition of total orderings and its application to knowledge discovery.

Then, we proceed to present a way of efficiently working with antichains and intersection of antichains, which can be very useful, especially in case of large sets of data. The last section details a series of experiments where it is shown that pattern structures can be implemented with an efficient intersection operation and that they have a generally better behavior than scaled contexts.

2 Pattern Structures for Structured Attributes

2.1 Pattern Structures

Formal Concept Analysis [7] can process only binary contexts. Pattern structures are an extension of FCA which allow a direct processing of such kind of data. The formalism of pattern structures was introduced in [1].

A *pattern structure* is a triple $(G, (D, \sqcap), \delta)$, where G is the set of objects, $(D, \sqcap)$ is a meet-semilattice of descriptions, and $\delta : G \to D$ maps an object to its description. In other words, a pattern structure composed of a set of objects, a set of descriptions equipped with a similarity operation denoted by $\sqcap$. This similarity operation is idempotent, commutative and associative. If $(G, (D, \sqcap), \delta)$ is a pattern structures then the derivation operators (Galois connection) are defined as:

$$A^\diamond := \prod_{g \in A} \delta(g) \qquad \text{for } A \subseteq G$$

$$d^\diamond := \{g \in G | d \sqsubseteq \delta(g)\} \qquad \text{for } d \in D$$

Each element in D is referred to as a *pattern*. The natural order on $(D, \sqcap)$, given by $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$ is called the subsumption order. Now a pattern concept can be defined as follows:

Definition 1 (Pattern Concept). *A pattern concept of a pattern structure $(G, (D, \sqcap), \delta)$ is a pair (A, d) where $A \subseteq G$ and $d \in D$ such that $A^\diamond = d$ and $A = d^\diamond$, where A is called the concept extent and d is called the concept intent.*

A pattern extent corresponds to the maximal set of objects A whose descriptions subsume the description d, where d is the maximal common description

for objects in A. The set of all pattern concepts is partially ordered w.r.t. inclusion on extents, i.e., $(A_1, d_1) \leq (A_2, d_2)$ iff $A_1 \subseteq A_2$ (or, equivalently, $d_2 \sqsubseteq d_1$), making a lattice, called pattern lattice.

2.2 Two original propositions on structured attribute sets

We briefly recall two original propositions supporting the present study. The first work is firstly published by Carpineto & Romano in [2] and then developed in [3]. The second work is related to the definition of pattern structures by Ganter & Kuznetsov in [1].

In [2, 3], the authors consider a formal context (G, M, I) and an extended set of attributes $M^* \supset M$ where attributes are organized within a subsumption hierarchy according to a partial ordering denoted by $\leq_{M^*}$. The following condition should be satisfied:

$\forall g \in G, m_1 \in M, m_2 \in M^* : [(g, m_1) \in I, m_1 \leq_{M^*} m_2] \Longrightarrow (g, m_2) \in I$

The subsumption hierarchy can be either a tree or an acyclic graph with a unique maximal element, as this is the case of attributes lying in a thesaurus for example. Then the building of a concept lattice from such a context can be done in two main ways. A first is to use a scaling and to complete the description of an object with all attributes implied by the original attributes. We discuss this scaling operation in detail later. The problem would be the space necessary to store the scaled context, especially in case of big data. A second way is to use an "extended intersection operation" between sets of attributes which is defined as follows. The intersection of two sets of attributes Y_1 and Y_2 is obtained by finding for each pair $(m_1, m_2), m_1 \in Y_1, m_2 \in Y_2$, the most specific attributes in M^* that are more general than m_1 and m_2, and then retaining only the most specific elements of the set of attributes generated in this way. Then if (X_1, Y_1) and (X_2, Y_2) are two concepts, we have:

$(X_1, Y_1) \leq (X_2, Y_2) \Longleftrightarrow \forall m_2 \in Y_2, \exists m_1 \in Y_1, m_1 \leq_{M^*} m_2$

In other words, this intersection operation corresponds to the intersection of two antichains as this is explained in [1], where the authors define the formalism of pattern structures and take as an instantiation structured attribute sets. More formally, it is assumed that the attribute set $(M, \leq_M)$ is finite and partially ordered, and that all attribute combinations that can occur must be order ideals (downsets) of this order. Then, any order ideal O can be described by the set of its maximal elements; $O = \{x | \exists y \in M,\ x \leq y\}$. *It should be noticed that the order considered on the attribute sets in [1] is reversed with respect to the order considered in [2, 3].* However, we keep the original definitions used in [1] in the present paragraph. These maximal elements form an antichain, and conversely, each antichain is the set of maximal elements of some order ideal. Thus, the semilattice $(D, \sqcap)$ of patterns in the pattern structure consists of all antichains of the ordered attribute set. In addition, it is isomorphic to the lattice of all order ideals of the ordered set, and thus isomorphic to the concept lattice of the context $(P, P, \not\geq)$. For two antichains AC_1 and AC_2, the infimum $AC_1 \sqcap AC_2$ consists of all maximal elements of the order ideal:

$\{m \in P \mid \exists ac_1 \in AC_1,\ \exists ac_2 \in AC_2,\ m \leq ac_1 \text{ and } m \leq ac_2\}$.

There is a "canonical representation context" (or an associated scaling operator) for the pattern structure $(G, (D, \sqcap), \delta)$ related to structured attribute sets, which is defined by the set of "principal ideals $\downarrow p$" as follows: (G, P, I) with $(g, p) \in I \Longleftrightarrow p \leq \delta(g)$.

In the next section, we make precise and discuss the pattern structure for structured attribute sets by taking the point of view of filters and not of ideals in agreement with the order from [2, 3], with the most general attributes above.

2.3 From Structured Attributes to Tree-shaped Attributes

An important case of structured attributes is "tree-shaped attributes", i.e., when the attributes are organized within a partial order corresponding to a rooted tree. If it is the case, then the root of the tree, denoted by $\top$, can be matched against the description of any object, while the leaves of this tree are the most detailed descriptions. For example, the root can correspond to the attribute 'Animal' and a leaf can correspond to the attribute 'Cat'; somewhere in between there could be attribute 'Mammal'.

An example of such kind of data naturally appears in the domain of semantic web data. For example, Figure 1 gives a small part of ACCS[1]. This attribute tree will be used as a running example and should be read as follows. If an object belongs to class C_1 (and probably to some other classes), then it necessarily belongs to classes C_{10}, C_{12}, and $\top$, e.g., if an object is a cat, then it is a mammal and an animal. Accordingly, the description of an object can include several classes, e.g., classes C_1, C_5 and C_8. Thus, some of the tree-shaped attributes can be omitted from the description of an object. However, they should be always taken into account when computing the intersection between descriptions. Thus, in order to avoid redundancy in the descriptions, we can allow only antichains of the tree as possible elements in the set D of descriptions, and then, accordingly compute the intersection of antichains.

An efficient way of computing intersection of antichains is explained in the next section. Here it is important to notice that although it is a hard task to efficiently compute intersection of antichains in an arbitrary partial order of attributes, the intersection of antichains in a tree can help in computing this more general intersection. Indeed, in a partial order of attributes, we can add an artificial attribute $\top$ that can be matched against any description. Then, instead of considering an intersection of antichains in an arbitrary poset we can take a spanning tree of it with $\top$ taken as the root. Although we have lost some relations between attributes, and, thus, the size of the antichains is probably larger, we can apply the efficient intersection of antichains of tree discussed below.

2.4 On Computing Intersection of Antichains in a Tree

In this subsection we show how to efficiently solve the problem of intersection of antichains in a tree. The problem is formalized as follows. A partial order is

[1] https://www.acm.org/about/class/2012

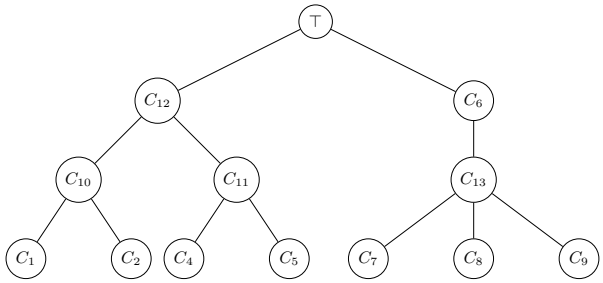


Fig. 1: A small part from ACM Computing Classification System (ACCS).

described by the Hasse diagram corresponding to the tree. The root is denoted by $\top$ and it is larger w.r.t. the partial order than any other element in the tree. Given a rooted tree $\mathcal{T}$ and two antichains X and Y, we should find an antichain Z such that (1) for all $x \in X \cup Y$ there is $z \in Z$ such that $x \leq z$ and (2) no $z \in Z$ can be removed or changed to $\tilde{z} < z$ without violating requirement (1).

If the cardinality of antichains X and Y is 1 then this task is reduced to the well-known problem of a Least Common Ancestor (LCA). In 1984 it was already shown that the LCA problem can be reduced to Range Minimum Query (RMQ) problem [8]. Later several simpler approaches were introduced for solving the LCA problem. Here we briefly introduce the reduction of LCA to RMQ in accordance with [9].

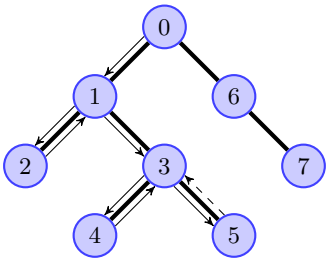
Reduction of LCA to RMQ. Given an array of numbers, the RMQ problem consists in efficient answering queries on the position of the minimal value in a given range (interval) of positions for this array. For example, given an array

Array [2 1 0 3 2]
Positions 1 2 3 4 5

where the first value is in position 1 and the last value is in position 5, the answer to the query on the position of the minimal number in the range 2–4, i.e., the corresponding part of array is [1;0;3], is 3 (the value of the 3rd element in the array is 0 and it is the minimal value in this range). Accordingly, the position of the minimal number in the range 1–2 (the part of the array is [2;1]) is 2. The good point about this problem is that it can be solved in $O(n)$ preprocessing computational time and in $O(1)$ computational time per one query [9], where n is the number of elements in the array.

In order to introduce the reduction of LCA to RMQ we need to know what is the depth of a tree vertex. The *depth* of a vertex in a rooted tree is the number of edges in the shortest path from that vertex to the root of the tree.

We create the array of depths of the vertices in the tree that is used as an input array for RMQ. We build this array in the following way. We traverse the tree in depth first order (see Figure 2). Every time the algorithm considers a



Depth array $\mathcal{D}$	[	0	1	2	1	**2**	3	**2**	3	**2**	1	0	**1**	2	**1**	0]
Corresponding vertex		v0	v1	v2	v1	**v3**	v4	**v3**	v5	**v3**	v1	v0	**v6**	v7	**v6**	v0
Positions		1	2	3	4	**5**	6	**7**	8	**9**	10	11	**12**	13	**14**	15

Fig. 2: Reducing RMQ task to LCA. Arrows show the depth first order traversal. The depth array $\mathcal{D}$ is accompanied by the corresponding vertices and positions.

vertex, i.e., the first visit or a return to the vertex, we should put the depth of that vertex at the end of the depth array $\mathcal{D}$. We also keep track of a vertex corresponding to each depth in $\mathcal{D}$. The depth array $\mathcal{D}$ has $2|\mathcal{T}|-1$ values, where $|\mathcal{T}|$ is the number of vertices in the tree.

Now for any value in $\mathcal{D}$ we know the corresponding vertex of the tree and any vertex of the tree is associated with several positions in $\mathcal{D}$. For example, in Figure 2 the value in the first position of $\mathcal{D}$, i.e., $\mathcal{D}[1]$, is 0, corresponding to the root of the tree. If we take vertex 3, then the associated values of $\mathcal{D}$ are on positions 5, 7, and 9.

Given two vertices $A, B \in \mathcal{T}$, let a be one of the positions in $\mathcal{D}$ corresponding to vertex A, let b be one of the positions in $\mathcal{D}$ corresponding to B. Then it can be shown that the vertex corresponding to the minimal value in $\mathcal{D}$ in the range a–b is the least common ancestor of A and B. For example, to find LCA between vertices 3 and 6 in Figure 2, one should first take two positions in $\mathcal{D}$ corresponding to vertices 3 and 6. Positions 5,7, and 9 in array $\mathcal{D}$ correspond to vertex 3, positions 12 and 14 correspond to vertex 6. Thus, we can query RMQ for ranges 5–14, 7–14, 7–12, etc. The minimal value in $\mathcal{D}$ for all these ranges is 0 located at position 11 in $\mathcal{D}$, i.e., $\texttt{RMQ}(5, 14) = 11$. Thus, the vertex corresponding to position 11, i.e., vertex 0, is the least common ancestor for vertices 3 and 6.

Let us notice that if $A \in \mathcal{T}$ is an ancestor of $B \in \mathcal{T}$ and a and b are two positions corresponding to the vertices A and B, then the position $\texttt{RMQ}(a, b)$ in $\mathcal{D}$ always corresponds to the vertex A, in most of the cases $\texttt{RMQ}(a, b) = a$. Thus we are also able to check if a vertex of $\mathcal{T}$ is an ancestor of another vertex of $\mathcal{T}$.

Now we know how to solve the LCA problem in $O(|\mathcal{T}|)$ preprocessing computational time and $O(1)$ computational time per query. Let us return to the problem of intersecting antichains of a tree.

Antichain intersection problem. Let us first discuss the naive approach to this problem. Given two antichains $A, B \subset \mathcal{T}$, one can compute the set

$\mathcal{D}$	[0	1	2	3	2	3	2	1	2	3	2	3	2	1	0	1	2	3	2	3	2	3	2	1	0]
	$\top$	C_{12}	C_{10}	C_1	C_{10}	C_2	C_{10}	C_{12}	C_{11}	C_4	C_{11}	C_5	C_{11}	C_{12}	$\top$	C_6	C_{13}	C_7	C_{13}	C_8	C_{13}	C_9	C_{13}	C_6	$\top$
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Fig. 3: Depth array, the corresponding vertices, and indices for the tree in Figure 1.

$\{\texttt{LCA}(a,b) \mid \forall a \in A \text{ and } \forall b \in B\}$. Then this set should be filtered for removing the comparable elements in order to get an antichain. It is easy to see that the result is the intersection of A and B but it requires at least $|A|\cdot|B|$ operations.

Let us reformulate this naive approach in terms of RMQ. Given a depth array $\mathcal{D}$ and two sets of indices $A, B \subseteq \mathbb{N}_{|\mathcal{D}|}$ forming an antichain, we should compute the set $Z = \{\texttt{RMQ}(a,b) \mid \forall a \in A \text{ and } \forall b \in B\}$ and then remove all elements $z \in Z$ such that there is $x \in Z \setminus \{z\}$ with the position $\texttt{RMQ}(z,x)$ corresponding to the same vertex as z, i.e., elements z corresponding to an ancestor of another element from Z.

Let us consider for example the tree $\mathcal{T}$ given in Figure 1. Figure 3 shows the depth array, the corresponding vertices, and indices of this array. Let us show how to compute the intersection of $A = \{C_1, C_5, C_8\}$ and $B = \{C_1, C_7, C_9\}$. The expected result is $\{C_1, C_{13}\}$. First we translate the sets A and B to the indices in array $\mathcal{D}$ for RMQ, i.e., $A = \{4, 12, 20\}$ and $B = \{4, 18, 22\}$. Then we compute RMQ for all pairs from A and B:

$$\{\texttt{RMQ}(4,4) = 4, \texttt{RMQ}(4,18) = 15, \texttt{RMQ}(4,22) = 15, \cdots, \texttt{RMQ}(20,18) = 19, \cdots\}.$$

Now we should remove positions corresponding to ancestors in the tree, e.g., $\texttt{RMQ}(4,15) = 15$ and, hence, 15 should be removed. The result is $\{4, 13\}$ representing exactly $\{C_1, C_{13}\}$.

Let us discuss two points that help us to reduce the complexity of the naive approach. Consider the positions $i \leq l \leq m \leq j$ and $k = \texttt{RMQ}(i,j)$, $n = \texttt{RMQ}(l,m)$. Then the depth in the position k is not larger than the depth in the position n, $\mathcal{D}[k] \leq \mathcal{D}[n]$. Hence the position $\texttt{RMQ}(k,n)$ corresponds to the same vertex as position k. For example, in Figure 3 $\texttt{RMQ}(4,6) = 5$ and $\texttt{RMQ}(2,7) = 2$. The value in position 5 in the array $\mathcal{D}$ is $\mathcal{D}[5] = 2$. It is larger than the value in position 2, $\mathcal{D}[2] = 1$. Thus, the value in position returned by RMQ for the larger range is smaller than the value in position returned by RMQ for the smaller range.

Thus, given two sets of indices $A, B \subseteq \mathbb{N}_{|\mathcal{D}|}$ corresponding to antichains, we can modify the naive algorithm by ordering the set $A \cup B$ and computing RMQ only for consecutive elements from different sets, rather then for all pairs from different sets. For example, for intersecting $A = \{4, 12, 20\}$ and $B = \{4, 18, 22\}$, we join them to the set $Z = \{4_A, 4_B, 12_A, 18_B, 20_A, 22_B\}$. Then, we compute RMQ only for consecutive elements from different sets, i.e., $\texttt{RMQ}(4,4) = 4$, $\texttt{RMQ}(4,12) = 8$, $\texttt{RMQ}(12,18) = 15$, $\texttt{RMQ}(18,20) = 19$, and $\texttt{RMQ}(20,22) = 21$. The cardinality of $A \cup B$ is less then $|A| + |B|$, hence, the number of the consecutive elements is $O(|A|+|B|)$, and, thus, the number of RMQs of consecutive elements is $O(|A| + |B|)$.

However, the set Z of RMQs of consecutive elements does not not necessarily correspond to an antichain in $\mathcal{T}$. Thus we should filter this set, in order to remove all ancestors of another elements form Z. Accordingly, it is clear that to filter the set Z it is enough to check only consecutive elements of Z. For example, the intersection of $A = \{4, 12, 20\}$ and $B = \{4, 18, 22\}$ gives us the following set $Z = \{4, 8, 15, 19, 21\}$. Let us now check the RMQs of consecutive elements. $\texttt{RMQ}(4, 8) = 8$, thus, 8 is an ancestor of 4 and 8 can be removed. Since 8 is removed, we compare $\texttt{RMQ}(4, 15) = 15$, thus, 15 should be also removed. Then we compute $\texttt{RMQ}(4, 19) = 15$, i.e., the indices 4 and 19 are not ancestors and both are kept. Now we compute $\texttt{RMQ}(19, 21) = 19$ and, thus, 19 should be removed (actually positions 19 and 21 correspond to the same vertex C_{13} and one of them should be removed). Thus, the result of intersecting A and B is $\{4, 21\}$ corresponding to the antichain $\{C_1, C_{13}\}$.

Since the number of elements in the set Z is $O(|A| + |B|)$, then overall complexity of computing intersection for two antichains $A, B \subset \mathcal{T}$ of a tree $\mathcal{T}$ is $O(|A| + |B|)$ or, taking into account that the cardinality of an antichain in a tree is less then the number of leaves (vertices having no descendants) in this tree, the complexity of computing intersection of two antichains is $O(|\texttt{Leaves}(\mathcal{T})|)$.

Antichain intersection by scaling. An equivalent approach for computing intersection of antichains is to scale the antichains to the corresponding filters. A *filter corresponding to an antichain* in a poset is the set of all elements of the poset that are larger then at least one element from the antichain. For example, let us consider a tree-shaped poset in Figure 1. A filter corresponding to the antichain $\{C_1, C_5, C_8\}$ is the set of all ancestors of all elements from the antichain, i.e., it is equal to $\{C_1, C_{10}, C_{12}, \top, C_5, C_{11}, C_8, C_{13}, C_6\}$.

The set-intersection of filters corresponding to the given antichains is a filter corresponding to the antichain resulting from intersection of the antichains. However this approach has a higher complexity. Indeed, the size of a filter is $O(|\mathcal{T}|)$ and, thus, the computational complexity of intersecting two antichains by means of a scaling is $O(|\mathcal{T}|)$ which is harder then $O(|\texttt{Leaves}(\mathcal{T})|)$ for intersecting antichains directly. Indeed, the number of leaves in a tree can be dramatically smaller than the number of vertices in this tree. For example, the number of vertices in Figure 1 is 13, while the number of leaves is only 7. Thus, the direct intersection of antichains is more efficient than the intersection by means of a scaling procedure.

Relation to intersection of antichains in partially ordered sets of attributes. As it was mentioned in the previous section, the intersection of antichains in arbitrary posets can be reduced to the intersection of antichains in a tree. However, the size of the antichain representing a description of an object can increase. Indeed, since we have reduced a poset to a tree, some relations have been lost, and thus the attributes that are subsumed in the poset for a given antichain A are no more subsumed in the tree for A, and hence should be added to A. However, the reduction is still more computationally efficient than

Table 1: Results of the experiments with different kind of data.

#objects is the number of objects in the corresponding dataset. #attributes is the number of numerical attributes before scaling. $|G|$ is the number of objects used for building the lattice. $|\mathcal{T}|$ is the size of the attribute tree and the number of attributes in the scaled context $|M|$. $\texttt{Leaves}(\mathcal{T})$ is the number of leaves in the attribute tree. $|\mathcal{L}|$ is the size of the concept lattice for the corresponding data. $t_{\mathcal{T}}$ is the computational time for data represented as a set of antichains in the attribute tree. $t_{\mathbb{K}}$ is the computational time represented by a scaled context, i.e., by a set of filters in the attribute tree; '*' shows that the we are not able to build the whole lattice. t_{num} is the computational time for numerical data represented by an interval pattern structure.

(a) Real data experiments.

Dataset	$\lvert G\rvert$	$\lvert\mathcal{T}\rvert$	$\texttt{Leaves}(\mathcal{T})$	$\lvert\mathcal{L}\rvert$	$t_{\mathcal{T}}$	$t_{\mathbb{K}}$
DBLP	5293	33207	33198	10134	45 sec	21 sec
Biomedical Data	63	1490	933	1725582	145 sec	162 sec

(b) Numerical data experiments.

Dataset	#objects	#attributes	$\lvert G\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\texttt{Leaves}(\mathcal{T})\rvert$	$\lvert\mathcal{L}\rvert$	$t_{\mathcal{T}}$	$t_{\mathbb{K}}$	t_{num}
BK	96	5	35	626	10	840897	37 sec	42 sec*	19 sec
LO	16	7	16	224	26	1875	0.043 sec	0.088 sec	0.024 sec
NT	131	3	131	140	6	128624	3.6 sec	6.8 sec	3.1 sec
PO	60	16	22	1236	58	416837	49 sec	57 sec*	10.7 sec
PT	5000	49	22	4084	60	452316	50 sec	38 sec*	15 sec
PW	200	11	94	436	21	1148656	60 sec	49 sec*	48 sec
PY	74	28	36	340	53	771569	46 sec	40 sec*	21 sec
QU	2178	4	44	8212	8	783013	28 sec	30 sec*	15.4 sec
TZ	186	61	31	626	88	650041	58 sec	43 sec*	22 sec
VY	52	4	52	202	15	202666	5.9 sec	11.6 sec	3 sec

computing the intersection of antichains in a poset by means of a scaling as it is discussed in the previous paragraph. However, for the reduction it could be interesting to find the spanning tree with the minimal number of leaves. Unfortunately, this is an NP-complete task and it thus cannot be applied for increasing the computational efficiency [10]. We should notice here that there is some work that solves the LCA problem for more general cases, e.g., lattices [11] or partially ordered sets [9]. However, it is an open question whether these works can help to efficiently compute intersection of antichains in the corresponding structures.

3 Experiments and Discussion

Several experiments are conducted using publicly available data on a MacBook with a 1.3GHz Intel Core i5, 4GB of RAM running OS X Yosemite 10.3. We have used `FCAPS`[2] software developed in C++ for dealing with different kinds of pattern structures. It can build a concept lattice starting from a standard formal context or from object descriptions given as antichains of a given tree. The last one is based on the similarity operation that is discussed above.

We performed our experiments on two datasets from different domains namely DBLP and biomedical data. In these datasets, object descriptions are given as subsets of attributes. A taxonomy of the attributes is already known based on domain knowledge. We compute a concept lattice in two different ways. In the first one, we directly compute the concept lattice from the antichains in a taxonomy. In the second one we scale every description to the corresponding filter of the taxonomy. After this we do not rely on the taxonomy and process the scaled context with standard FCA.

The first data set is DBLP, from which we extracted a subset of papers with their keywords published in conferences in Machine Learning domain. The taxonomy used for classifying such kind of triples is ACM Computing Classification System (ACCS)[3].

The second data set belongs to the domain of life sciences. It contains information about drugs, their side effects (SIDER[4]), and their categories (DrugBank[5]). The taxonomies related to this dataset are MedDRA [6] for side effects and MeSH[7] for drug categories.

The parameters of the datasets and the computational results are shown in Table 1a. It can be noticed that for DBLP the context consists of 5293 objects and 33207 attributes, in the taxonomy of the attributes we have 33198 leaves meaning that most of attributes are mutually incomparable. It took 45 seconds to produce a lattice having 10134 concepts directly from the descriptions given by antichains of the taxonomy. To produce the same lattice starting from a scaled context the program only takes 21 seconds. However, if we consider the biomedical data, the approach based on antichains is better. Indeed, it takes 145 seconds, while the computation starting from the scaled contexts takes 162 seconds. In this case, the dataset contains 1490 attributes with 933 leaves. Thus, the direct approach works faster if the number of leaves is significantly smaller than the number of vertices. It is worth noticing that the size of antichains is significantly smaller than the size of the filters, and thus our approach is more efficient. However, when the number of leaves is comparable to the number of vertices, our approach is slower. Although in this case our approach has the same

[2] https://github.com/AlekseyBuzmakov/FCAPS
[3] https://www.acm.org/about/class/2012
[4] `http://sideeffects.embl.de/`
[5] `http://www.drugbank.ca/`
[6] http://meddra.org/
[7] http://www.ncbi.nlm.nih.gov/mesh/

computational complexity as the scaling approach, the antichain intersection problem requires more efforts than the set intersection.

Since the efficiency of the antichain approach is high for the trees with a low number of leaves, we can use this method to increase efficiency of standard FCA for special kind of contexts. In a context (G, M, I) an attribute m_1 can be considered as an ancestor of another attribute m_2 if any object containing the attribute m_2 also contains the attribute m_1. Accordingly we can construct an attribute tree $\mathcal{T}$ and rely on it for computing intersection operation. In this case the set of attributes M and the set of vertices of $\mathcal{T}$ are the same and $|M| = |\mathcal{T}|$. The second part of the experiment was based on this observation.

We used numerical data from Bilkent University in the second part of the experiments[8]. It was converted to formal contexts by the standard interodinal scaling. The scaled attributes are closely connected, i.e., there are a lot of pairs of attributes (m_1, m_2) such that the set of objects described by m_1 is a subset of objects described by m_2, i.e., $(m_1)' \subseteq (m_2)'$. Thus, we can say that $m_1 \leq m_2$. Using this property we built attribute trees from the scaled contexts. These trees have many more vertices than leaves, thus, the approach introduced in this paper should be efficient. We compare our approach with the scaling approach. Moreover, recently, it was shown that interval pattern structures (IPS) can be efficiently used to process such kind of data [12]. Accordingly we also compared our approach with IPS.

The results are shown in Table 1b. Compared to Table 1a it has several additional columns. First of all, since for numerical data we typically got large lattices, in most of the cases we considered only part of the objects. The actual number of used objects is given in the column $|G|$, while the total size of the dataset is given in the column '#objects', e.g., `BK` dataset contained 96 objects, while we have used only 35. In addition for every dataset we also provide the number of the numerical attributes, e.g., `BK` has 5 numerical attributes. We should notice that when we built the lattice from some datasets by standard FCA, the lattice was so large that the memory was swapping and we stopped the computation. It was not the case for our approach since antichains requires less memory to store than the corresponding filters. The fact of swapping is shown by '*' next to computational time in column $t_{\mathbb{K}}$. In addition we also show the time for IPS to process the same dataset. For example, the processing of `BK` dataset took 37 seconds by our approach, took more than 42 seconds by standard FCA and memory had started swapping, and took 19 seconds by IPS.

This experiment shows that our approach takes not only less time to compute concept lattice, but also requires less memory, since there is no memory swapping. We can also see that the computation time for IPS is smaller than for our approach. However, IPS is only applicable for numerical data, while our approach can be applied for all cases when attributes of a context are structured. For example, we can deal with graph data scaled to the set of frequent subgraphs where many such attributes are subgraphs of other attributes.

[8] `http://funapp.cs.bilkent.edu.tr/DataSets/`

4 Conclusion

In this paper we recalled two approaches for dealing with structured attributes and explained how we can compute intersection of antichains in tree-shaped posets of attributes, an essential operation for working with structured attributes. Our experiments showed the computational efficiency of the proposed approach. Accordingly, we are interested in applying our approach to other kinds of data such as graph data. Moreover, the generalization of our approach to other kinds of posets is also of high interest.

References

1. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: ICCS. LNCS 2120, Springer (2001) 129–142
2. Carpineto, C., Romano, G.: A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning **24**(2) (1996) 95–122
3. Carpineto, C., Romano, G.: Concept Data Analysis: Theory and Applications. John Wiley & Sons, Chichester, UK (2004)
4. Alam, M., Napoli, A.: Interactive exploration over RDF data using Formal Concept Analysis. In: International Conference on Data Science and Advanced Analytics, DSAA 2015, Paris, France, October 19 - October 21, 2015, IEEE (2015)
5. Caspard, N., Leclerc, B., Monjardet, B.: Finite Ordered Sets. Cambridge University Press, Cambridge, UK (2012) First published in French as "Ensembles ordonnés finis : concepts, résultats et usages", Springer 2009.
6. Pichon, E., Lenca, P., Guillet, F., Wang, J.W.: Un algorithme de partition d'un produit direct d'ordres totaux en un nombre fini de chaînes. Mathématiques, Informatique et Sciences Humaines **125** (1994) 5–15
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin/Heidelberg (1999)
8. Gabow, H.N., Bentley, J.L., Tarjan, R.E.: Scaling and Related Techniques for Geometry Problems. In: Proc. Sixt. Annu. ACM Symp. Theory Comput. STOC '84, New York, NY, USA, ACM (1984) 135–143
9. Bender, M.A., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P.: Lowest common ancestors in trees and DAGs. J. Algorithms **57**(2) (2005) 75–94
10. Salamon, G., Wiener, G.: On finding spanning trees with few leaves. Inf. Process. Lett. **105**(5) (2008) 164–169
11. Aït-Kaci, H., Boyer, R., Lincoln, P., Nasr, R.: Efficient Implementation of Lattice Operations. ACM Trans. Program. Lang. Syst. **11**(1) (January 1989) 115–146
12. Kaytoue, M., Kuznetsov, S.O., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in formal concept analysis. Inf. Sci. (Ny). **181**(10) (2011) 1989 – 2001

Author Index

Editors:	Sadok Ben Yahia, Jan Konecny
Title:	CLA 2015, Proceedings of the Twelfth International Conference on Concept Lattices and Their Applications
Publisher & Print:	UBP, Limos, Campus Universitaire des Cézeaux, 63178 AUBIERE CEDEX – FRANCE
Place, year, edition:	Clermont-Ferrand, France, 2015, 1st
Page count:	xiii+254
Impression:	50
Archived at:	`cla.inf.upol.cz`

Not for sale

ISBN 978–2–9544948–0–7