Léonard Kwuida
Barış Sertkaya (Eds.)

LNAI 5986

Formal Concept Analysis

8th International Conference, ICFCA 2010
Agadir, Morocco, March 2010
Proceedings

Springer

Lecture Notes in Artificial Intelligence 5986

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

Subseries of Lecture Notes in Computer Science

Léonard Kwuida Barış Sertkaya (Eds.)

Formal Concept Analysis

8th International Conference, ICFCA 2010
Agadir, Morocco, March 15-18, 2010
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Léonard Kwuida
Zurich University of Applied Sciences, School of Engineering
Technikumstrasse 9, 8401 Winterthur, Switzerland
E-mail: kwuida@gmail.com

Barış Sertkaya
SAP Research Center Dresden
Chemnitzer Strasse 48, 01187 Dresden, Germany
E-mail: baris.sertkaya@sap.com

Library of Congress Control Number: 2010921136

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

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-11927-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-11927-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180 5 4 3 2 1 0

Preface

This volume contains selected papers presented at ICFCA 2010, the 8th International Conference on Formal Concept Analysis. The ICFCA conference series aims to be the prime forum for dissemination of advances in applied lattice and order theory, and in particular advances in theory and applications of Formal Concept Analysis.

Formal Concept Analysis (FCA) is a field of applied mathematics with its mathematical root in order theory, in particular the theory of complete lattices. Researchers had long been aware of the fact that these fields have many potential applications. FCA emerged in the 1980s from efforts to restructure lattice theory to promote better communication between lattice theorists and potential users of lattice theory. The key theme was the mathematical formalization of *concept* and *conceptual hierarchy*. Since then, the field has developed into a growing research area in its own right with a thriving theoretical community and an increasing number of applications in data and knowledge processing including data visualization, information retrieval, machine learning, sofware engineering, data analysis, data mining in Web 2.0, analysis of social networks, concept graphs, contextual logic and description logics.

ICFCA 2010 took place during March 15–18, 2010 in Agadir, Morocco. We received 37 high-quality submissions out of which 17 were chosen as regular papers in these proceedings after a competitive selection process. Less mature works that were still considered valuable for discussion at the conference were collected in the supplementary proceedings. The papers in the present volume cover advances in various aspects of FCA ranging from its theoretical foundations to its applications in numerous other fields. In addition to the regular papers, this volume also contains four keynote papers arising from the seven invited talks given at the conference. We are also delighted to include a reprint of Bernhard Ganter's seminal paper on his well-known algorithm for enumerating closed sets. The high quality of both regular and keynote papers was the result of the hard work of the authors, invited speakers and the reviewers.

We wish to thank the members of the Program Committee and the members of the Editorial Board for their involvement that ensured the scientific quality of this volume. We would also like to thank the external reviewers for their valuable comments. Additional thanks go to the organization Committee and to the *Institut Supérieur d'Informatique Appliquée et de Management (ISIAM)* for sponsoring the conference and hosting it. Finally we wish to thank the Conference Chair Rokia Missaoui and her team for the tremendous amount of work they put before and during the conference to make it a real success.

March 2010

Léonard Kwuida
Barış Sertkaya

Organization

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

Executive Committee

Conference Chair

Rokia Missaoui	Université du Québec en Outaouais, Canada

Local Chair

Ameur Boujenoui	Telfer School of Management, University of Ottawa, Canada

Conference Organizing Committee

Omar Abahmane	Université du Québec en Outaouais, Canada
Souad Bennani	Institut Supérieur d'Informatique Appliquée et de Management (ISIAM), Agadir, Morocco
Lahcen Boumedjout	Université du Québec en Outaouais, Canada
Aziz Bouslikhane	Institut Supérieur d'Informatique Appliquée et de Management (ISIAM), Agadir, Morocco
Jamal Elachmit	Institut Supérieur d'Informatique Appliquée et de Management (ISIAM), Agadir, Morocco

Conference Proceedings

Program Chairs

Léonard Kwuida	Zurich University of Applied Sciences, School of Engineering, Winterthur, Switzerland
Barış Sertkaya	SAP Research Center Dresden, Germany

Jürg Schmid	Universität Bern, Switzerland
Andreja Tepavčević	University of Novi Sad, Serbia
Petko Valtchev	Université du Québec à Montréal, Canada
Vilem Vychodil	Binghamton University - State University of New York, USA
Serhyi Yevtushenko	Luxoft, Ukraine

External Reviewers

Mikhail A. Babin	Higher School of Economics, Moscow, Russia
Loïc Cerf	INSA Lyon, France
Heiko Reppe	Technische Universität Dresden, Germany
Chris Schulz	University of Pretoria, South Africa
Branimir Seselja	University of Novi Sad, Serbia
Gregor Snelting	Karlsruhe Institute of Technology, Germany
Fritz Venter	University of Pretoria, South Africa
Denny Vrandecic	Karlsruhe Institute of Technology, Germany

Table of Contents

Invited Talks

Regular Contributions

History

About the Enumeration Algorithms of Closed Sets

Alain Gély[1], Raoul Medina[2], and Lhouari Nourine[2]

[1] Université Paul Verlaine - Metz
IUT de Metz - Département STID
Ile du Saulcy
57045 Metz Cedex 1
alain.gely@univ-metz.fr
[2] Université Blaise Pascal – LIMOS
Campus des Cézeaux,
63173 Aubière Cedex, France
{medina,nourine}@isima.fr

Abstract. This paper presents a review of enumeration technics used for the generation of closed sets. A link is made between classical enumeration algorithms of objects in graphs and algorithms for the enumeration of closed sets. A unified framework, the *transition graph*, is presented. It allows to better explain the behavior of the enumeration algorithms and to compare them independently of the data structures they use.

1 Introduction

Generation algorithms of combinatorial objects is an important problem in data mining, artificial intelligence and discrete mathematics. The combinatorial objects considered in this paper are maximal cliques of a graph, maximal bicliques of a bipartite graph and closed itemsets of a binary relation. Indeed, any binary relation can be seen as a bipartite graph where concepts are maximal bicliques of this graph.

There exists several algorithms which generates maximal bicliques of a bipartite graph using polynomial space [2,4,11,17,27] . For instance, Next-Closure [11] uses linear space to enumerate the closed sets of a closure operator (this algorithm is more general than the problem of enumerating all maximal bicliques of a bipartite graph since it can be applied with any closure operator). In formal concept analysis, several algorithms which construct the concept lattices (or Galois lattice) of a context have been proposed such as Lindig algorithm [20]. Data mining domain produced numerous algorithms for the enumeration of closed itemsets of a collection of transactions [23,24,28,29,31,32,33]. The family of closed itemsets is usually smaller than the family of itemsets: the search space is thus reduced.

Given a bipartite graph $G = (U, V, E)$, all algorithms in [2,11,29,23] have a worst case complexity of $O(|U|^2 \times |V|)$ per maximal biclique: all those algorithms are polynomial delay. However, their practical behaviour may differ. Since they

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 1–16, 2010.

are used in different domains where the amount of data is quite important, some requirements on their practical behaviours are expected. Several comparative studies, based on benchmark data, have been made in [9,18,31].

Those experimental studies effectively show that there exist behavior differences between these algorithms. This leads to the following question: where does the difference come from since they all have the same theoretical time complexity? Corollary questions might be: Do some particular data structures allow to improve efficiency? Is the enumeration order significant? Are the theoretical complexities over estimated for some algorithms?

One can already notice that it might be possible that all these aspects might interact. How to know where the efficiency difference comes from? From the data structures they use? From the order in which they generate the objects?

We propose a generic approach to the enumeration problem which allows us to overcome the differences in data structures and concentrate on the enumeration order of the maximal bicliques. This approach relies on the transition graph $H(G) = (\mathcal{C}(G), T)$ of maximal cliques of a graph G, where $\mathcal{C}(G)$ is the set of maximal cliques of G and $(C, C') \in T$ if there is a transition function from C to C'. Given two maximal cliques C and C' of G, we say that a transition exists between C and C' if C' can be obtained from C using a simple transformation which we will define later.

We show that classical algorithms can be seen as searches of this transition graph. The differences are explained by the way of testing if an edge of $H(G)$ belongs to the covering tree defined by the algorithm as well as by the way this covering tree is searched. We will define the covering trees and searches associated by the two algorithms of Johnson et al[17] and Tsukiyama et al[27].

We will study the impact of the choice of transitions to consider in the covering tree. By modifying the way of choosing an element, a more or less important number of edges of $H(G)$ will never be tested. This is equivalent to finding a partial graph of $H(G)$ with less edges than $H(G)$ but allowing to reach all maximal bicliques of G [13].

Simulating a search of $H(G)$ can be done in several ways using the same data structure. We will see that, using the same data structure, significant differences of behavior might appear which might explain the difference of behavior of known algorithms. Once those differences put forward, we will try to analyze them in order to find which particular properties might explain them. Once those properties are identified, they can be used to improve existing algorithms (or improve their analysis), but also to improve the way usual benchmarks are described.

In this paper, we chose the formalism of graph theory in order to show the link between the enumeration of maximal cliques, maximal bicliques and closed itemsets.

We first recall some basis on enumeration problems and their associated complexities. Then we focus on the enumeration of closed sets by using the link between maximal cliques, maximal bicliques and closed itemsets.

2 Enumeration Problems

In this paper we will focus on the maximal cliques enumeration problem which is stated as follows:

Problem: Maximal Cliques Enumeration
Instance: A graph $\mathcal{G} = (V, E)$.
Question: Generate all maximal cliques of G.

In this section, we recall some principle on enumeration problems: their link with decision problems, their complexities and common technics used for enumerating objects.

2.1 From Decision Problems to Enumeration Problems

Complexity analysis of enumeration problems takes into account the size of the output. This differs from classical optimization problems which have an output size identical to the input size. In the case of enumeration problems, the size of the output is usually exponential in the size of the input. As a consequence, the complexity of algorithms solving enumeration problems will be exponential. In order to analyze and classify these problems, the size of the output is also taken into account. This is clearly the case when considering the decision problem associated to an enumeration problem.

For instance, the decision problem associated to the Maximal Cliques Enumeration problem is stated as follows:

Problem: Maximal Cliques of a graph
Instance: A graph $\mathcal{G} = (V, E)$ and a collection $\mathcal{H} \subseteq 2^V$.
Question: Does $\mathcal{C}(G) = \mathcal{H}?$, where $\mathcal{C}(G)$ is the set of maximal cliques of G

The complexity of enumerating maximal cliques depends on the complexity of its associated decision problem. Usually, generating a solution as well as testing if an object is a solution is supposed to be done in polynomial time. Often, the decision problem is in CO-NP. For instance, it is easy to test if a subset of V is a maximal clique of G which do not belong to $\mathcal{H}$. Thus, the main difficulty of an enumeration problem is to check if *all* objects have been generated (i.e. to test if $\mathcal{C}(G) \subseteq \mathcal{H}?$).

If the decision problem is polynomial then the algorithm which solves it will return either an answer "yes" or a solution which does not belong to $\mathcal{H}$. We thus add this new solution to $\mathcal{H}$ and apply the decision problem solver again until we find the complete set of maximal cliques. This gives an *enumeration algorithm* which has a polynomial time complexity.

The related counting problem is to compute the number of maximal clique of a given graph. Clearly there is no connection between enumeration problems and their associated counting problem. For example, the problem of counting the number of maximal cliques of a graph is #P-complete while the enumeration problem is polynomial (see [19,30] for counting problems).

2.2 Complexity Analysis

In this section, we recall the time complexities associated to enumeration algorithms. We consider an enumeration problem with input data of size n and the total number of objects to enumerate is N.

Polynomial time complexity. An enumeration algorithm has a *p*olynomial time complexity if its time complexity is in $O((n+N)^k)$, with k a constant.

Polynomial time complexity per object. An algorithm is said to have a *polynomial time complexity per object* if its time complexity is in $O(n^k \times N)$, with k a constant.

In other words, when dividing the overall time complexity of the algorithm by the number of generated objects we obtain an order of magnitude which is polynomial in the size of the input. Such time complexity is also called *amortized polynomial time complexity*.

In a similar way, an algorithm is said to have a *linear time complexity per object* if its overall complexity is in $O(n \times N)$. An algorithm with an overall complexity of $O(N)$ is said to have an amortized constant complexity.

Be careful however: having a polynomial time complexity per object does not necessarily mean that between two consecutive objects we have a polynomial time.

Note that algorithms with polynomial time complexity and polynomial time complexity per object are also called output polynomial or polynomial total time [17].

Polynomial delay complexity algorithms. A *polynomial delay complexity* algorithm is an algorithm which has the following properties:

- Computation of the first object is done in polynomial time according to the size n of the input.
- Between two consecutive enumerated objects there is a polynomial time according to the size n of the input.
- After the computation of the last object, the algorithm halts in polynomial time according to the size n of the input.

Any enumeration algorithm which is polynomial delay is also polynomial. The polynomial delay property is interesting when enumerating a list of objects. Indeed, if a process has to be applied on each generated object, then a pipeline process can be put in place: the first generated objects are processed while there are still objects to generate. Such pipeline process cannot be applied on enumeration algorithms which do not have this polynomial delay property.

Imposed output order enumeration. Another interesting property for enumeration algorithms is to enumerate the objects in a specific order: for instance according to the lexicographic order, according to the size of the objects, etc.

This property is sometimes incompatible with the polynomial delay property. For instance, Johnson et al [17] proved that it is not possible to have a polynomial delay algorithm which enumerates the maximal cliques of a graph in the reverse lexicographic order (unless $P = NP$).

For similar reasons, it is not possible to have a polynomial delay algorithm which enumerates the maximal cliques in a descending order of their sizes (unless $P = NP$).

Polynomial space complexity. We distinguish two families of enumeration algorithms: those requiring a polynomial space and those requiring an exponential space.

Algorithms requiring an exponential space may benefit from stored informations which is not the case for polynomial space algorithms. This avoids to redo some computations and obtain a better time complexity. For instance, for the problem of enumerating the closed sets of a closure system, the best algorithm using an exponential space [22] has a time complexity which is quadratic per enumerated objects (but it is not polynomial delay) while the best algorithms using a polynomial space have a time complexity which is cubic per enumerated object (and it is also polynomial delay).

2.3 Enumeration Technics

Main idea of the different enumeration technics is to guarantee that when enumerating combinatorial objects, each object is enumerated only once. This section presents different classical enumeration technics.

Lexicographic order. The first idea that arises when one has to enumerate objects is to define a total order among those objects. If such a total order can be found, then it is always possible to define a lexicographic order (the dictionary order) on these objects.

Once the total order is chosen, we can associate to each object a function $NEXT()$ which returns the next object to enumerate in the lexicographic order.

The time complexity of an algorithm based on a lexicographic order depends on the computation of the first object and on the cost of the $NEXT()$ function.

Gray code. A Gray code is a way of enumerating the objects such that the difference between two consecutive objects is "small". The exact definition of "small" difference depends on the context. Let $\mathcal{C}$ be a collection of combinatorial objects and f a proximity relation defined over $\mathcal{C}$. A Gray code of $\mathcal{C}$ exists if and only if $G = (\mathcal{C}, f)$ admits an hamiltonian path or circuit. Here, the difficulty is to prove the existence of such hamiltonian path and then to find an algorithm which searches this path (for a survey on Gray codes see [25]).

Backtrack. The backtrack enumeration technique comes from the divide and conquer paradigm. The idea is to split a family $\mathcal{F}$ of objects to enumerate in two object families $\mathcal{F}_\pi$ and $\mathcal{F}_{\overline{\pi}}$, according to a property π, such that:

– Objects in $\mathcal{F}_\pi$ satisfy the property π.
– Objects in $\mathcal{F}_{\overline{\pi}}$ do not satisfy the property π.

The same process can be recursively applied to split the families $\mathcal{F}_\pi$ and $\mathcal{F}_{\overline{\pi}}$ in smaller families until having families of size 1 (or families that can be easily enumerated).

Backtrack algorithms face two problems:

– Splitting the family $\mathcal{F}$ in two might break the structure of the family. For instance, if we split a lattice in two there is no particular reason to obtain two lattices.
– Testing, at each step, if a family $\mathcal{F}$ contains an object to generate might be difficult. Should the test be polynomial, one would obtain immediately a polynomial algorithm.

3 Closed Sets Enumeration

In this section, we consider the problem of enumerating all closed sets: the time complexity is thus dependent on the number of closed sets.

Given a set X, this enumeration is done using a closure operator usually defined over an implication cover or a context. Usually, when the main focus is on algorithmic properties, either the reduced context (i.e. the context corresponding to the family of irreducible elements of $\mathcal{F}$) or a non redundant (possibly minimum) family of implications are considered.

In the following, we focus on algorithmic methods for the enumeration of maximal bicliques of a bipartite graph (in other words, concepts of a context). We study those algorithms using a larger framework: the enumeration of the maximal cliques of an arbitrary graph. Indeed, we will show that from this problem it is straightforward to reduce to the maximal bicliques enumeration problem.

This general framework also allows us to put forward similarities existing in methods that were discovered independently.

3.1 Definitions

A graph or undirected graph $G$is a pair $G = (V, E)$, where V is a finite set of vertices and E is a set of edges. The neighborhood of a vertex v is denoted by $\Gamma(v) = \{u \in V s.t.\ uv \in E\}$.

Definition 1. *A clique of $G = (V, E)$ is a set of vertices $C \subseteq V$, such that for any v_1 and v_2 in C, the edge (v_1, v_2) belongs to E.*

A clique C of G is maximal if and only if for any $x \in V \setminus C$, $C \cup x$ is not a clique.

The set of all maximal cliques of G is denoted by $\mathcal{C}(G)$. When no ambiguity is possible, we simply denote this family by $\mathcal{C}$. Whenever a total order on maximal cliques is necessary, we denote C_i the i^{th} clique of $\mathcal{C}(G)$.

Definition 2. *A bipartite graph G, denoted by $G = (U, V, E)$ is a graph $(U \cup V, E)$ such that U and V are stable sets.*

Definition 3. *Let $G = (U, V, E)$ be a bipartite graph. A biclique of G is a pair (X, Y), $X \subset U$ and $Y \subset V$ such that for any $x \in X$, $y \in Y$ we have $(x, y) \in E$.*

A biclique (X, Y) is said maximal if, for any $x \in (U \setminus X)$ (resp. $y \in (V \setminus Y)$), the set $(X \cup \{x\}, Y)$ (resp $(X, Y \cup \{y\})$) is not a biclique.

If a biclique (X, Y) is not maximal, we say that it is absorbed by another biclique (X_1, Y_1) (denoted by $(X, Y) \subseteq (X_1, Y_1)$) if and only if ($X \subseteq X_1$ and $Y \subseteq Y_1$) or ($X \subseteq Y_1$ and $Y \subseteq X_1$).

Previous definitions match the definitions of a context (a relation between objects and attributes) and of a concept (a maximal biclique). The notion of implication appears with the absorption of a biclique by another biclique.

The set of maximal bicliques of G is denoted by $\mathcal{B}(G)$.

Problem transformation. The problem of enumerating the maximal bicliques of a bipartite graph $G = (U, V, E)$ is equivalent to the problem of enumerating the maximal cliques of $G' = (U \cup V, E')$, where $E' = E \cup \{(x, y) \mid x, y \in U \text{ and } x \neq y\} \cup \{(x, y) \mid x, y \in V \text{ and } x \neq y\}$.

This transformation, used for instance in [21] consists of replacing the two stables U and V by cliques: maximal cliques of G' correspond to maximal bicliques of G.

3.2 A Formalism for the Enumeration of Cliques: The Transition Graph

In this section, we introduce the transition graph of the maximal bicliques of a graph. This structure allows to extend and to unify several algorithms and extend results of [21]. While in [21] the focus was on efficiency, our aim is to have a unified framework in order to compare the behavior of enumeration algorithms.

Let G be a graph, we denote by $H(G) = (\mathcal{C}(G), T)$ the (oriented) transition graph of maximal cliques of G. Vertices of $H(G)$ are maximal cliques of G and its edges are *possible transition* between two cliques. Given two maximal cliques C and C' of G, there exists a transition between C and C' if C' can be computed from C by a simple transformation (the transition function which will be defined below).

3.3 A Lexicographic Transition Function

The definition of the graph $H(G)$ requires to choose a transition function. If no particular knowledge on properties of the graphs is given, the transition function can only use lexicographic properties (information on the labels of the vertices).

Moreover, the transition function should be chosen such that the transition graph $H(G)$ has "good" properties. In particular, since we want to use this graph as the support for the enumeration, it is necessary that this graph be strongly connected in order to reach all maximal cliques during the search of the graph.

Consider a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$. Given a clique C of G and a vertex $i \notin C$, we denote $K_i(C)$ the lexicographically smaller maximal clique induced by the vertices $(C \cap \{1, \ldots, i-1\} \cap \Gamma(i)) \cup \{i, \ldots, n\}$.

This set of vertices corresponds to the common prefix from 1 to $i-1$ between C and $\Gamma(i)$ extended with vertices greater or equal to i. One can notice that vertices lower to i form a clique, but the completion of this clique with vertices greater than i is not necessarily a maximal clique of G.

There are two cases:

- $K_i(C)$ is a *maximal* clique of the graph G.
- $K_i(C)$ is a *non maximal* clique of the graph G. In this case, one can notice that the vertices that need to be added in order to obtain a maximal clique are necessarily lower to the vertex i which are not in C.

Given two cliques C and C' of G, a transition is possible between C and C' if there exists a vertex $i \notin C$ such that $K_i(C) = C'$. In other words, there exists a transition from C to C' if $K_i(C)$ is the lexicographically smaller maximal clique of G having $C \cap \{1, 2, \ldots, i-1\} \cap \Gamma(i)$ as prefix.

The operator K_i, using only lexicographic and simple operations (intersection, clique completion), is the one we use as *transition function* in the remaining of the paper.

The usage of lexicographic properties is frequent and is at the core of Next-Closure [11], a simple and efficient algorithm for the enumeration of closed sets. The use of the transition graph allows to compare the behavior (as well as the properties) of Next-closure [11] and Tsukiyama et al [27] algorithm for the enumeration of maximal cliques.

3.4 Transition Graph Properties

Given a graph $G = (V, E)$, we define the transition graph $H(G) = (\mathcal{C}(G), T)$ of maximal cliques of G by :

1. $\mathcal{C}(G)$ the set of maximal cliques of G
2. (C, C') belongs to T if there exists a transition between C and C' using K_i.

Consider C and C' two maximal cliques such that there exists a transition from C to C'. Then there exists a unique $i \notin C$ such that $C' = K_i(C)$. Indeed, for $i < j$, it is impossible to have i in $K_j(C)$ since $i \notin C$.

Since the vertex i is unique, we associate it to each edge (C, C') allowing the transition from C to C', denoted by $Label(C, C') = i$.

Example 1. *Consider the graph G in figure 7. Note that the graph $H(G)$ is not symmetric since $(245, 123) \in T$ and $(123, 245) \notin T$.*

We recall some properties of the transition graph already proved in [14,15]:

- $H(G)$ is strongly connected
- C_0, the lexicographically smallest maximal clique can be reached by transition from any maximal clique

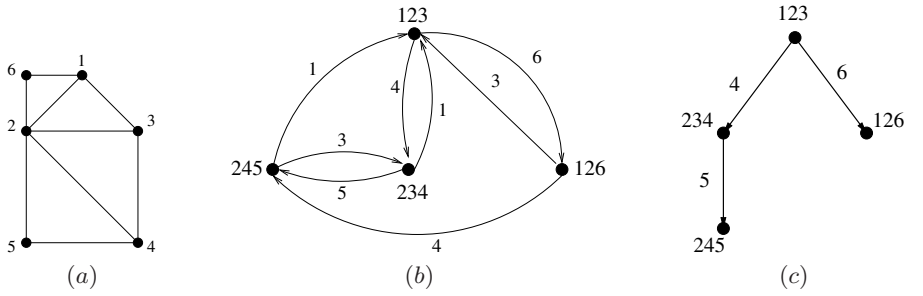


Fig. 1. (*a*) A graph G. (*b*) The transition graph of maximal cliques of G. (*c*) A covering tree of (*b*).

- A maximal clique admits at most n outgoing edges with n the number of vertices of G.
- Computation of a transition can be done in $O(m)$, with m the number of edges of G.

Those properties allow us to consider any search of $H(G)$ as the enumeration of maximal cliques of a graph. As a consequence, the time complexity of the enumeration depends on the number of transitions searched in order to reach once all maximal cliques, i.e. $O(mn)$ by maximal clique (in the worst case, all transitions are searched with a cost of $O(m)$ per transition). We recognize here the classical complexities associated to these problems.

Moreover, classical algorithms for enumerating the maximal cliques can be seen as particular instances of searches of the transition graph [15], simplifying their study and comparison.

- The algorithm of Johnson et al [17], which enumerates the maximal cliques in the lexicographic order with a polynomial delay time complexity but using an exponential space, corresponds to a search which is very close to a breadth first search (explaining the exponential space required).
- The algorithm of Tsukiyama et al [27], which enumerates the maximal cliques in no particular order, using a polynomial delay time complexity and using a polynomial space, corresponds to a depth first search (explaining the polynomial space required).

These two algorithms search a covering tree of the transition graph defined as follows:

Let $T(G) = (\mathcal{C}(G), T_b)$ be a partial graph of $H(G)$ such that $(C, C') \in T_b$, with $Label(C, C') = 1$, if and only if

1. $C' = Ki(C)$,
2. $C' \cap 1, \ldots, i$ is a maximal clique in G_i, the restriction of G to its first i vertices,

3. C is the lexicographically smallest maximal clique of G containing $C \cap \{1, \cdots, i-1\}$.

This covering tree has an interesting property: the sequence of labels of a path from C_0 (the lexicographically smallest maximal clique) to any other clique C is ascending (see Fig. 1(c) for an example of such covering tree).

3.5 Transition Graph and Enumeration of Maximal Bicliques

We have seen that the problem of enumerating the maximal bicliques of a bipartite graph can be reduced to the problem of enumerating the maximal cliques of a graph. In this section, we study this particular case. Figure 2 shows an example of transformation of a bipartite graph (a) into a graph (b) such that the maximal bicliques of (a) are exactly the maximal cliques of (b).

In the remaining of the paper, by hypothesis, we suppose that vertices of G' are ordered such that given $v \in V$ and for any $u \in U$, we have $v < u$. In other

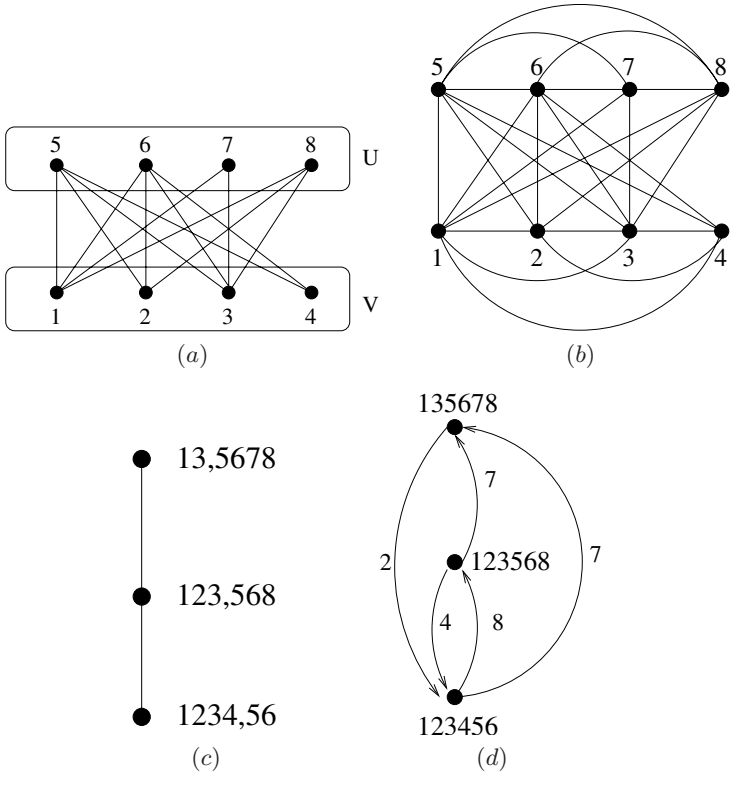


Fig. 2. (a) A bipartite graph G; by hypothesis, vertices of U have labels lexicographically greater than those of vertices of V. (b) the graph G'. (c) concept lattice of G. (d)$H(G')$.

words, any vertex of V precede any vertex of U in the order. This property allows us to take advantage of the lexicographic properties which arise in numerous enumeration algorithms and to exhibit similarities between the enumeration of maximal cliques and the enumeration of maximal bicliques.

Under the labeling hypothesis of the vertices of U and V, we show that the covering graph of the concept lattice of G is a partial graph of $H(G)$. This allows to consider enumeration algorithms based on a search of this lattice as simple particular cases of searches of the transition graph.

Property 1. *Let $C_0, C_1, \ldots, C_k$ be a path of $H(G')$ such that $Label(C_i, C_{i+1}) \in U$ for all $0 \leq i < k$. Then for any $u \in U$, $u \in C_i$ implies $u \in C_{i+1}$, i.e. $C_1 \cap U \subseteq C_2 \cap U \subseteq \ldots \subseteq C_k \cap U$.*

Proof. Let $u \in C_i \cap U$ with $0 \leq i < k$. We show that $u \in C_{i+1} \cap U$. Suppose that $Label(C_i, C_{i+1}) = l$ and $C_{i+1} = K_l(C_i)$.

If $u \leq l$ then $u \in C_i \cap \{1, \ldots, l\}$ and thus $u \in K_l(C_i)$.

If $u > l$. Suppose that $u \notin C_{i+1}$. Then there exists $x \in C_{i+1}$ such that $(u, x) \notin E$. Clearly, $C_{i+1} \cap V \subseteq C_i \cap V \Rightarrow x \notin V$.

Thus $x \in U$, leading to a contradiction since for any $x, y \in U$, we have $(x, y) \in E$. □

Recall that given two maximal bicliques $B_1 = (X_1, Y_1)$ and $B_2 = (X_2, Y_2)$, we have $B_1 < B_2$ in the concept lattice if and only if $X_1 \subseteq X_2$ (or, respectively $Y_2 \subseteq Y_1$).

By definition, the maximal cliques of a graph are incomparable relatively to the inclusion. We define an order on the maximal cliques in order to recover the natural order of maximal bicliques of a bipartite graph.

Definition 4. *Let C and C' be two maximal cliques of a graph G'. We say that C is lesser than C', denoted by $C <_{\subseteq} C'$, if $C \cap U \subseteq C' \cap U$.*

We denote by $\prec_{\subseteq}$ the covering relation.

Next proposition shows that if a maximal biclique covers another maximal biclique in the concept lattice then there exists a transition in $H(G')$ between the corresponding maximal cliques.

Proposition 1. *Let $H(G) = (\mathcal{C}(G'), T)$ be the transition graph of maximal cliques of $G' = (U \cup V, E')$ and C, C' two maximal cliques of G' such that $C \prec_{\subseteq} C'$. Then $(C, C') \in T$.*

Proof. Recall that a maximal biclique of a bipartite graph G is uniquely determined by one of its parts. In the same way, we can uniquely determine a maximal clique of G' by the set of its vertices in U (resp. V).

Since $C \prec_{\subseteq} C'$, we have $C \cap U \subseteq C' \cap U$ and $C' \cap V \subseteq C \cap V$. Let $X = (C' \cap U) \setminus (C \cap U)$ and i the smallest vertex of X. Then we have:

- $(i, j) \in E'$ for all $j \in C \cap U$ (by construction).
- $(i, j) \notin E'$ for all $j \in (C' \setminus C) \cap V$. Indeed, let $C'' = K_i(C)$. Then $C'' \neq C$ since $j \in C''$ and $j \notin C'$. Moreover, we have $C' \cap V \subset C'' \cap V \subset C \cap V$. This implies $C \not\prec_{\subseteq} C'$

Thus $C \cap V \cap \Gamma(i) = C' \cap V$ et $C' = K_i(C)$. □

From proposition 1, we deduce that the covering graph of the concept lattice of a bipartite graph G is a partial graph of $H(G')$, the transition graph of maximal cliques of G'. Moreover, note that this covering graph uses only labels from only one of the two sets of vertices of the bipartite graph. This allows us to consider any enumeration algorithm based on the concept lattice as a particular instance of a search of $H(G')$.

3.6 Maximal Bicliques Enumeration - Insights

A classical enumeration algorithm for closed sets using a closure operator is Next-closure [11]. It enumerates all closed sets using a variation of the lexicographic order called lectic order. We recall the definition of this lectic order which is also used by the datamining algorithm LCM [28].

Definition 5. *Let A and B be two sets of elements totally ordered. The set A is lectically lower than the set B (denoted by $A <_{lec} B$) if the smallest differing element belongs to B.*

More formally, there exists $i \in B \backslash A$ such that $A \cap \{1, 2, \ldots, i-1\} = B \cap \{1, 2, \ldots, i-1\}$

For instance, the family of sets $\{a,\ ab,\ b,\ c,\ cd,\ ce\}$ (given in the classical lexicographic order) will be ordered $\{c,\ ce,\ cd,\ b,\ a,\ ab\}$ in the lectic order.

The lectic order, if it differs on the way sets containing a particular element are enumerated, remains an order based on the labels of the vertices. In the lexicographic order, sets containing the smallest element are first enumerated, while in the lectic order sets that *do not contain* this element are first enumerated.

A classical presentation of Next-Closure is the one given by algorithm 1. In this algorithm, we denote by $A <_i B$ if A is lectically lower than B and the smallest difference is the element i. This implementation has the advantage to be concise and easy to implement.

On line 3 of the algorithm 1, one can notice the similarity with the algorithm of Tsukiyama et al [27] on maximal cliques:

- Next-closure computes the closure of $(A \cap \{1, 2, ..., i-1\}) \cup i$
- [27] tries to complete $(A \cap \{1, 2, ..., i-1\}) \cup i$ to obtain a maximal clique.

Whenever Next-Closure rejects a set because of an inappropriate label, Tsukiyama algorithm will test a non valid transition in the transition graph.

Next proposition shows that the algorithm of Tsukiyama et al enumerates maximal bicliques of a bipartite graph in the lectic order over U.

Algorithm 1. Next-closure algorithm

Data: J a set of elements
C(), a closure operator over J
$A \subseteq J$ a closed set
Result: B, the closed set which is the immediate successor of A in the lectic order.

```
begin
1   | K = J\A, sorted by ascending order
2   | for i ∈ K do
3   |   | B ← C((A ∩ {1, 2, ..., i − 1}) ∪ i)
4   |   | if X <_i B then
    |   |   | Return B
    |   | end
    | end
end
```

Proposition 2. *The algorithm of Tsukiyama et al [27] enumerates maximal bicliques of* $G = (U, V, E)$ *in the lectic order of elements in* U.

Proof. Let G' constructed from G by transforming U and V as cliques and $T(G')$ the covering tree of $H(G')$ used by the algorithm of Tsukiyama.

Recall that, by hypothesis, $v < u$ for all $v \in V$, $u \in U$.

Let C and C' be two maximal cliques such that there exists a path between C and C' in $T(G')$ and $u \in U$. The sequence of labels from C to C' is ascending. Moreover, we can use only labels of U and we have the property that if $u \in C$, then $u \in C'$.

If we restrict to U, whenever the algorithm considers a clique, all maximal cliques containing vertices of U with smaller labels have already been enumerated, which corresponds to the lectic order. □

Indeed, Next-Closure and Tsukiyama algorithm use the same basis which is the lectic enumeration. Applied to the enumeration of maximal cliques of a graph G' constructed from the bipartite graph G, the two algorithms are identical.

Next-Closure is the basis of numerous studies which aim to adapt or extend it. For instance, we can cite the work of Huaigo Fu and Engelbert Mephu Nguifo [9,10,8] around Scaling-Next-closure, a parallel version of Next-closure, the work of R. Emilion, G. Lambert and G. Lévy [6] to apply Next-closure to non binaries relations, allowing the use of Next-Closure on maximal and stochastic lattices which are an extension of concept lattice [5].

The strength of Next-Closure (simplicity of the implementation, arbitrary closure operator, enumeration of closed sets in a particular order without the need to store them, ...) is also its major drawback, in its original version, since it cannot take advantage of supplementary informations. Particularly, by using lexicographic properties rather than structural properties, the efficiency of Next-Closure depends on the labeling of the elements.

There exist methods to re-label the set J in order to gain some practical efficiency. In [12], the authors show that it is more efficient to label the elements in such a way that for $i, j \in J$ and given a closure operator $C()$, if $C(i) < C(j)$, then $i > j$.

For instance, with such a labeling, Next-Closure does not compute any useless closed set in the case of distributive lattices. The algorithm has then a complexity of $O(m)$ per closed set for such lattices.

Still in [12] the authors show that the complexity of Next-closure becomes $O(\omega(U).m)$ per closed set when considering a chain decomposition of U, where $\omega(U)$ is the width of the order $(U, \leq)$.

All the above mentioned ameliorations come from a knowledge easily workable: the knowledge of some implications. Testing and rejecting a closure in Next-Closure relatively to a vertex i means that there exists a vertex j, $j < i$, whose label should not appear. This is equivalent to, when applying the operator $K_i()$ on this closed set, not obtaining a maximal clique (the element j will not be present); the transition is not valid.

This information is given by the order induced by the elements of the bipartite graph and can be seen as an implication $i \rightarrow j$.

Knowing unitary implications (those having a single element as antecedent) allows to:

- sort the elements in order to avoid that an element which implies another one has a greater label.
- not testing all transitions, but only those for which no element with a smaller label is implied.

It is this property which is used in algorithms such as [2,20] in order to avoid testing all possible labels for the transitions. In this sense, those algorithms do no explore all the possible transitions of $H(G')$.

A good knowledge of structural properties of a closure system (lattices classes, filter notion implications, etc...) could lead to amelioration of classical algorithms of graph theory for the enumeration of maximal cliques. Following these two tracks (historical and structural) should allow a better understanding of the behavior of the algorithms and, in a second step, consider ameliorations.

4 Future Work

In this paper, we use $H = (\mathcal{C}(G), T)$ the transition graph of maximal cliques of a graph G defined in [15]. We therefore showed the link between lattice algorithms and classical graph algorithms following the work of A. Berry [1]. Next step is to study the properties of $H(G)$ for particular classes of graphs. Indeed, some graph classes have better time complexities for the enumeration of maximal cliques (see for instance [3]). Does $H(G)$ have additional structural properties in such cases?

We have seen that $H(G)$ is not symmetrical, but that any clique can be reached from C_0. Does $H(G)$ have an hamiltonian path? Algorithms [17] and

[27] search only edges (C, C') with label i in $H(G)$ such that $C \cap \{1, \ldots, i-1\} \cap \Gamma(i) \cup \{i\}$ is a maximal clique of G_i. The transition graphs $H(G)$ has other paths. Does that allow to have an hamiltonian path (at least for particular classes of graphs)?

One can also notice that the transition graph of maximal cliques of G depends on the labeling of vertices of G. Can we find a "good" labeling of the vertices of G such that $H(G)$ has interesting properties? If this is the case, it might indicate that such labeling correspond to some properties of G that should be studied.

A refined complexity analysis of Next-Closure algorithm [11] shows that its time complexity is in $O(\omega(U).m)$ [12]. Another challenging problem would be to obtain an $O(m)$ algorithm using polynomial space.

Finally, this transition graph is also valid for the enumeration of stables or of minimal covers of edges. Can we extend it to other combinatorial objects such as the minimal transversals of an hypergraph? The answer to this questiondefined in seems to be difficult. Many work has been done in the FCA community [19,26] as well as in the graph theory community [7,16] without success for the moment.

References

1. Berry, A., McConnell, R.M., Sigayret, A., Spinrad, J.: Very Fast Instances for Concept Generation. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3874, pp. 119–129. Springer, Heidelberg (2006)
2. Bordat, J.P.: Calcul pratique du treillis de galois d'une correspondance. Math. Sci. Hum. 96, 31–47 (1986)
3. Cay, Y., Kong, M.C.: Generating all maximal cliques and related problems for certain perfect graphs. Congressus Numerantium 90, 33–55 (1992)
4. Chein, M.: Algorithme de recherche de sous-matrice première d'une matrice. Bull. Math. R. S. Roumanie 13 (1969)
5. Diday, E., Emilion, R.: Maximal and stochastic galois lattices. Discrete Applied Mathematics 127, 271–284 (2003)
6. Emilion, R., Lambert, G., Lévy, G.: Algorithms for general galois lattice building. Technical report, CERIA, Université Paris IX Dauphine (2001)
7. Fredman, M.L., Khachiyan, L.: On the complexity of dualization of monotone disjunctive normal forms. Journal of Algorithms 21(3), 618–628 (1996)
8. Fu, H.: Algorithmique des Treillis de concepts: Application á la fouille de données. PhD thesis, Université d'Artois, France (2005)
9. Fu, H., Fu, H., Njiwoua, P., Mephu Nguifo, E.: A comparative study of fca-based supervised classification algorithms. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 313–320. Springer, Heidelberg (2004)
10. Fu, H., Mephu Nguifo, E.: A parallel algorithm to generate formal concepts for large data. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 394–401. Springer, Heidelberg (2004)
11. Ganter, B.: Two basic algorithms in concept analysis. Technical report, Technische Hoschule Darmstadt (1984)
12. Ganter, B., Reuter, K.: Finding all closed sets: a general approach. Order 8 (1991)
13. Gely, A.: Algorithmique combinatoire: Cliques, Bicliques et systèmes implicatifs. PhD Thesis, Université Blaise Pascal, Clermont-Ferrand, France (2005)

14. Gély, A., Nourine, L.: Algorithmique d'énumération: Cliques, bicliques, itemset fermés. Revue I3: Information - Interaction - Intelligence, numéro Special (Juin 2007)
15. Gély, A., Nourine, L., Sadi, B.: Enumeration aspects of maximal cliques and bicliques. Discrete Applied Mathematics 157(7), 1447–1459 (2009)
16. Eiter, T., Gottlob, G., Makino, K.: New results n monotone dualization and generating hypergraph transversals. SIAM J. on Computing 32(2), 514–537 (2003)
17. Johnson, D.S., Yannakakis, M., Papadimitriou, C.H.: On generating all maximal independent sets. Information Processing Letters 27, 119–123 (1988)
18. Kuznetsov, S., Obiedkov, S.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental and Theoritical Artificial Intelligence (JETAI) 14(2/3), 189–216 (2002)
19. Kuznetsov, S., Obiedkov, S.: Some decision and counting problems of the Duquenne-Guigues basis of implications. Discrete Applied Mathematics 156(11), 1994–2003 (2008)
20. Lindig, C.: Fast concept analysis. In: Stumme, G. (ed.) Working with Conceptual Structures - Contributions to ICCS 2000, pp. 235–248 (2000)
21. Makino, K., Uno, T.: New algorithms for enumerating all maximal cliques. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 260–272. Springer, Heidelberg (2004)
22. Nourine, L., Raynaud, O.: A fast algorithm for building lattices. Information Processing Letters 71, 199–204 (1999)
23. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering Frequent Closed Itemsets for Association Rules. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1999)
24. Pei, J., Han, J., Mao, R.: Closet: an efficient mining algorithm for mining frequent closed itemsets. In: Proc. of ACM-SIGMOD, International Workshop on Data Mining and Knowledge Discovery (DMKD 2000), Dallas (May 2000)
25. Savage, C.: A survey of combinatorial (Gray) codes. SIAM Review 39(4), 605–629 (1997)
26. Sertkaya, B.: Some Computational Problems Related to Pseudo-intents. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 130–145. Springer, Heidelberg (2009)
27. Tsukiyama, S., Ide, M., Aiyoshi, M., Shirawaka, I.: A new algorithm for generating all the independent sets. SIAM J. Computing 6, 505–517 (1977)
28. Uno, T., Asai, T., Uchida, Y., Arimura, H.: Lcm: An efficient algorithm for enumerating frequent closed item sets. In: ICDM 2003 - Proc. of Workshop on Frequent Itemset Mining Implementations, FIMI (2003)
29. Uno, T., Kiyomi, M., Arimura, H.: Lcm ver. 2: Efficient mining for algorithms for frequent/closed/maximal itemsets. In: ICDM 2004 - Proc. of Workshop on Frequent Itemset Mining Implementations, FIMI (2004)
30. Valiant, L.: Lcm ver. 2: The complexity of Enumeration and Reliability Problems. SIAM Journal on Computing 8(3) (1979)
31. Wang, J., Han, J., Pei, J.: Closet+: Searching for the best strategies for mining frequent closed itemsets. In: Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington D.C. (August 2003)
32. Zaki, M.J., Hsiao, C.-J.: Charm: an efficient algorithm for closed itemset mining. In: 2nd SIAM International Conference on Data Mining, Arlington (April 2002)
33. Zaki, M.J., Hsiao, C.-J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Transaction on Knowledge and Data Engineering 17(4), 462–478 (2005)

Mathematics Presenting, Reflecting, Judging*

Rudolf Wille

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

Abstract. To understand what it means to present, to reflect, and to discuss mathematics, mathematics shall in the first place be characterized in connection of Peirce's classification of the inquiring sciences. The threefold view of the universal categories of Peirce suggests to orientate the expositions

- for presenting on the self-image of mathematics (as a First),
- for reflecting on the relationship of mathematics to the real world (as a Second), and
- for judging on the sense, meaning and connection of mathematics (as a Third).

These expositions support the following basic thesis: sense and meaning of mathematics finally lie in the fact, that mathematics may effectively support the rational communication of human beings.

Contents

1 Mathematics in the Context of the Inquiring Sciences

If one wants to understand better what it means to represent, to reflect, and to judge mathematics, then it does not suffice to take only mathematics under consideration, instead, one has also to see mathematics in its impact across its boundaries. In the frame of science such a view can be taken from the classification of the inquiring sciences which has been presented and explained by the American scientist and philosopher Charles Sanders Peirce in 1904 in his "Intellectual Autobiography" [Pe04] (see next page). Peirce understands this classification as a correction of Comte's classification of sciences of which he takes over above all the idea of an arrangement of the sciences according to the decreasing degree of abstraction. Peirce's classification of sciences is arranged in such a manner that each science

* This article is an English version of the German publication [Wi05b].

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 17–33, 2010.

- should be related in its general principals exclusively on the sciences which are erected above it, and
- should be related in its examples and special facts on the sciences which are presented below it.

For instance, the physical geometry refers to the 3-dimensional Euclidean vector space of mathematics as a general principal which allows every imaginable spatial form in deductive clearness ideal to construct (cf. [Hu92], p.19). Conversely, Mathematics uses the spatial structures of the physical geometry to concretize significantly existing mathematical structures and to abstract new theory promoting structures.

Classification of the inquiring sciences (Peirce)
I. mathematics
II. philosophy
  A. phenomenology
  B. normative science
    1. esthetics
    2. ethics
    3. logic
  C. metaphysics
III.special sciences

According to Peirce, mathematics is the most abstract science and the only hypothetic science. Mathematics "investigates solely the consequences of hypotheses, without considering that they correspond to anything real or not" (s. [Pe04], p.71f); here the consequences of hypotheses are understood as conditionals, where mathematics "does not take over responsibility for the truthness of the appertaining premise, however for the appertaining conclusions, which can be necessarily drawn out of such premises" (s. [Pe11]; p.458).

Characteristic for modern mathematics is, according to Peirce, that the research of mathematicians is founded on the same forms of thinking (as numbers, geometric figures, functions, and set structures), which makes possible for mathematicians to develop together a (coherent) cosmos of forms of potential reality. The ultimate aim, which mathematics is approaching in the long run, is for Peirce the discovery of the real potential world, in which the actual existence is nothing more as an arbitrary locality ([Pe92]; p.121).

Directly after mathematics Peirce lists the scientific philosophy as the most general science of the actual reality, which studies

- what is immediately present (phenomenology),
- what should be and what not (normative science),
- what is reality as regularity (metaphysics).

The phenomenology explores the universal qualities of phenomenons in its immediate character. The normative sciences generally treat the conformity of things with regard to its purposes:

- the esthetics view those thinks whose purposes incarnate qualities of feeling,
- the ethics view those thinks whose purposes lie in the action,
- the logic views those thinks whose purpose is to represent something.

For a deeper understanding of mathematics, in particular the insight in its relationship to the (philosophical) logic is necessary, because mathematics gains its relation to the world especially by the connection of mathematical and logical thinking (cf. [Wi01c]). This connection is based on the relation of potential and actual reality. Therefore, according to Peirce, logic can take over the given knowledge of mathematics in all its generality and base on it logical principles. For Peirce all necessary logical conclusions are mathematical conclusions, i.e. it is performed as an observation of something which is equivalent to a mathematical diagram ([Pe92]; p.116). How mathematical diagrams can represent mathematical conclusions, that Peirce has tried to show above all by his existential graphs [Pe93].

Also phenomenology, esthetics, and ethics are in fruitful connection to mathematics, in particular about their principles which are finding applications in logic. From phenomenology with its doctrine of the three universal categories of firstness, secondness, and thirdness Peirce deduces three kinds of logical conclusions: the abduction, the induction, and the deduction.

- The abduction creates out of the horizon of self-understanding a hypothesis as a First.
- The induction confirms a hypothesis about actual facts as a Second.
- The deduction concludes a hypothesis from valid premises on the basis of logical laws as a Third.

This means: the deduction proves that something must be the case; the induction shows that something is actually effective; and the abduction assumes that something might be the case.

Esthetics deliver for the logic determinations of that, what is a purpose in itself (as a First), admirable and desirable under all thinkable circumstances. The ethics clarifies for the logic which purposes (as a Second) a human may choose reasonably enough in different situations. Logic is therewith compelled to show which means of representation (as a Third) are available for the fulfillment of such purposes (cf. [Oe93]; p.113).

The universal categories of firstness, secondness, and thirdness are for Peirce the most general forms of thought of the phenomenology and therewith also of the philosophy, which makes the universal categories in particular meaningful for mathematics. Therefore their characterization shall be shortly given here:

- Category "the First" is the Idea of that which is such as it is regardless of anything else. That is to say, it is a Quality of Feeling.

- Category "the Second" is the Idea of that which is such as it is as being Second to some First, regardless of anything else and in particular regardless of any law, although it may conform to a law. That is to say, it is Reaction as Phenomenon.
- Category "the Third" is the Idea of that which it is as being a Third, or a Medium, between a Second and its First. That is to say, it is Representation as an element of the Phenomenon.

Understanding mathematics means, according to the universal categories of Peirce, to comprehend mathematics

- in the sense of firstness, i.e. how mathematics is, as itself, regardless of anything else,
- in the sense of secondness, i.e. how mathematics is, in relationship to the real world, regardless as anything else,
- in the sense of thirdness, i.e. how mathematics is, in its embedding in the real world.

These threefold understandings can be made explicit on the base of communication forms to present, to reflect, and to judge mathematics. With that , the concern about general mathematics can be also made more clear, by which it is important to obtain a more conscious self-understanding of mathematics, a better relationship to the world, and deeper senses, meanings, and connections about mathematical activities in total ([Wi01a]; p.3).

2 Presenting Mathematics (as a First)

The self-understanding of mathematics expresses itself in the respective teaching- and research-activities of mathematicians as well as in their productions. From basic meanings for mathematics, the respective abstract semantics have been extended in the 20th century from the number-, function-, figure-semantics to the semantics on set structures.

In special publications mathematics is presented in a restrictive conventionalized language which is, as a common language, considerably presented by linear sequences of signs and refers to the semantics of set structures. The restrictive conventions are results of the strive of mathematicians towards consent and coherence of greatest possible size; this has convincingly presented by Susanne Prediger in [Pr01]. How strongly those restrictions are usually be handled, that shall be shown by example 1.

The text of the example 1 describes the beginning of the second section of the research article "Restructuring lattice theory: an approach based on hierarchies of concepts" [Wi82]. This text presents the mathematical contents to be imparted by the conventions of the present mathematics, and is therefore founded on the semantics of set structures. There are linguistically defined by sets the basic mathematical notions: (formal) context, concept, extent, intent, subconcept, and superconcept and established the relation with the mathematical theory of Galois

connections, which is used with the takeover of a basic proposition of this theory. Also with the further expositions, the article follows established mathematical conventions, first of all the basic pattern of "definition - theorem - proof", and lays the foundation for a mathematical theory of concept lattices.

2. CONCEPT LATTICES

We start defining a *context* as a triple (G,M,I) where G and M are sets, and I is a binary relation between G and M; the elements of G and M are called *objects* [Gegenstände] and *attributes* [Merkmale], respectively. If gIm for $g \in G$ and $m \in M$ we say: the object g *has* the attribute m. For $A \subseteq G$ and $B \subseteq M$ we define

$$A' := \{m \in M \mid gIm \text{ for all } g \in A\},$$
$$B' := \{g \in G \mid gIm \text{ for all } m \in B\}.$$

The mappings given by $A \mapsto A'$ and $B \mapsto B'$ are said to form a *Galois connection* between the power sets of G and M, i.e. they fulfill the following basic properties (cf. Birkhoff [3; pp. 122-125]).

PROPOSITION. *For a context* (G,M,I):

(1)	$A_1 \subseteq A_2$	*implies*	$A_1' \supseteq A_2'$	*for* $A_1, A_2 \subseteq G$,
(1')	$B_1 \subseteq B_2$	*implies*	$B_1' \supseteq B_2'$	*for* $B_1, B_2 \subseteq M$,
(2)	$A \subseteq A''$	*and*	$A' = A'''$	*for* $A \subseteq G$,
(2')	$B \subseteq B''$	*and*	$B' = B'''$	*for* $B \subseteq M$.

Now, a *concept* [Begriff] of the context (G,M,I) may be defined as a pair (A,B) where $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$; A and B are called the *extent* and the *intent* of the concept (A,B), respectively. The hierarchy of concepts is captured by the definition

$$(A_1,B_1) \leq (A_2,B_2) :\Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_1 \supseteq B_2)$$

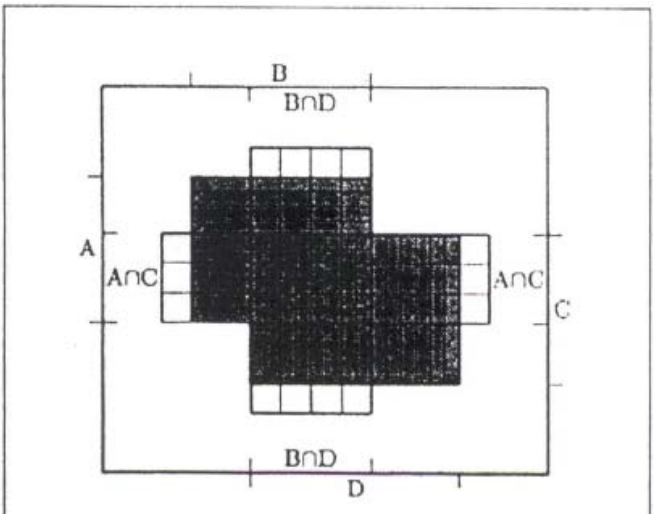
for concepts (A_1,B_1) and (A_2,B_2) of (G,M,I); (A_1,B_1) is called the *subconcept* of (A_2,B_2), and (A_2,B_2) is called the *superconcept* of (A_1,B_1).

Expl. 1: out of „Restructuring lattice theory: an approach based on hierarchies of concepts"

In university teaching and in expert lectures, mathematics is presented as a rule also in conventional technical language, which is however often supported by diagrams and pictures. The diagrams and pictures indicate that the human thinking, also that of mathematicians, is finally holistic. Example 2 shows a diagram of a formal context (G,M,I) with which generally for two formal concepts (A,B) and (C,D) of (G,M,I) their

- infimum $(A \cap C, (A \cap C)')$ (greatest common subconcept) and
- supremum $((B \cap D)', B \cap D)$ (smallest common superconcept)

can be visualized. Such "rectangle-diagrams" have been manifoldly proven in development and imparting of the mathematical theory of concept lattices (cf. [GW96]).



Example 2: Rectangle-Diagramms of formal concepts

In the expert communication of directly cooperating mathematicians, mathematics is presented diagrammatically in a considerable extent. In such a case the diagrammatic visualization follows rarely given conventions, but they follow most immediately out of the desire to make holistic figures of thinking graphically and with that more explicit. An impressive example comes from the well-known mathematician Zvonimir Janko: he developed for his research about finite simple groups elaborated diagram-representations of group-theoretical structure-connections, which made his discoveries explicit in such a way that his assistant could translate for Janko those diagrams in a conventional scientific language.

In the compartment-exceeding communication, a successful presentation of mathematics is possible only then, when the persons concerned become already accustomed in a field of mathematics and its semantics. Experiences show that such an acclimation is very time-consuming for non-mathematicians (cf. [Wi01b]). Therefore the educational socialization would absolutely be aimed in the fundamental semantics of mathematics - especially the ones of the 20th century. For acclimatizing in the actual semantics of set structures the entering with the theory of concept lattices has been proven extremely well both in university-like and also school-like courses and while doing so the multifarious possibilities of applications have motivated and convinced the learner.

3 Reflecting Mathematics (as a Second)

To reflect the connection of mathematics to the real world, this means first of all to understand mathematical thinking as abstraction of logical thinking as it is practised in mathematics. On such reflexions the research program of restructuring mathematics was based at the Technical University Darmstadt since 1978 (s. [Wi01b]). Under the up to now tackled restructurings of subareas, the restructuring of lattice theory has been proved most effectively. These restructuring is based on the mathematization of concept, the basic unit of logical thinking [Se01] as it is shown in the example 1. An extensive discussion of this mathematization can be read in [Wi05a]. In total it is astonishing how much lattice-theoretic thinking could be logically activated about the correspondence between the mathematical order relations and the logical subconcept-superconcept-relation based on formal contexts (s. for instance [Wi87], [SW00], [Wi00b], [Wi02a]).

Furthermore it is to reflect how new patterns of thinking can be generated with the growing potentials of mathematics, which mathematize present logical patterns. The most research of applied mathematics develops and secures mathematizations. This is also right for the theory of concept lattices as applied lattice theory. Thus, in the last decade the traditional logic with its doctrines of concept, judgment and conclusion as so-called "contextual logic" is successfully mathematized on the basis of the theory of concept lattices (s. [Wi96], [Wi97], [Pr98], [MSW99], [GW99], [Wi00a], [Wi03], [Da03], [Wi04a], [DK05]).

Finally, it has also to be reflected which mathematical structures and procedures can be used constructively in the real world to produce normatively logical relationships. Such relationships are first of all realized in systems as for instance in systems of numbers, measurements, technologies, and organizations (cf. [Fi88]). Mathematical structures and procedures are inserted in great diversity for construction and utilization of such systems. Also concept lattices, understood as mathematized concept systems, are applied many-sidedly in the real world, first of all to make logical connections transparent and rational (s. for instance [Ek04]).

An interesting type of application of concept lattices is the support of empirically founded theory building in the sense of [Ke94] and [SC96] (s. also [SWW01]). As an example of such an application, it shall be briefly described the concept analytic support of a Phd-project in musicology about the the theme "Simplicity. Genesis and change of a conceptual landscape in the music esthetics of the 18th century" (cf. [Ma00]). 270 historical sources served as a foundation of the theory building process, the content of which was derived with a normative vocabulary of more than 400 text attributes to the theme "simplicity". With the help of the obtained stock of data, a multitude of thematically determined concept lattices was formed as representatives of local parts of theories, which could be aggregated to larger theory pieces with the TOSCANA-Software [KSVW94] (developed for concept lattices). With such aggregations the thematical concept lattices were proved and eventually improved, therewith the theory process proceedingly wins and could reach to a convincing conclusion (cf. [MW99]).

The basic difference between the mathematical thinking of ideal objects and the logical thinking of real objects has always to be made consciously for all relations of mathematics to the real world. Otherwise it threatens a dangerous sense deflation as E. Husserl elaborated in his late writings about the "crisis of the European sciences" [Hu92]. For example, the objects of modern Euclidean geometry are "limit forms", hence ideal objects which are not allowed to be identified with real objects. Already Aristotle viewed space and time continua not consisting of points, but points form only boundaries ("limit forms") of continua. How this view can be reflected and explained is elaborated in [Wi04b].

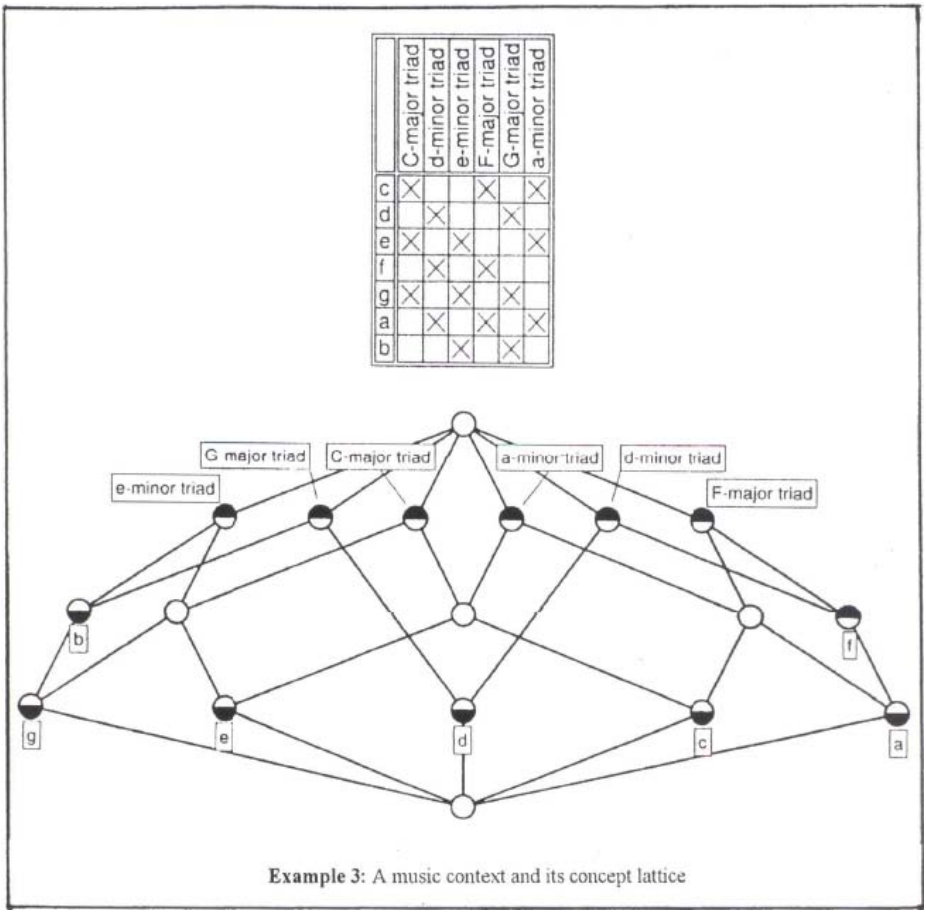
4 Judging Mathematics (as a Third)

Being embedded into the real world, this gives mathematics its actual sense and meaning. Therefore, judging mathematics has to be primarily related to the purposes which should be fulfilled by the logical realizations of mathematics. For judging mathematics the normative sciences, i.e. esthetics, ethics, and logic, are basic.

1. The esthetics guides to judge mathematical patterns of thinking and their logical realizations thereupon, how far they are themselves admirable and desirable in connection with the given purpose. In general, Peirce defines something as admirable ("esthetically good") if it consists of a manifold of parts, the connection of which gives the whole a simple positive quality of feeling ([Pe03b]; p.201). Such a quality of feeling is very confidential for Mathematicians; they experience again and again when a mathematical network of problems disperses into a right, transparent whole. Andrew Wiles has impressively described his breakthrough to the ultimate solution of Fermat's-Problem: "Suddenly, totally unexpectedly, I had this incredible revelation. Nothing I'll ever do again will ..." at the moment tears welled up and Wiles was choking with emotion. What Wiles realized at that fateful moment was "so indescribably beautiful, it was so simple and so elegant ... and I just stared in disbelief." To point out the high status of admire and of desire in mathematics, Paul Erdös spoke with pleasure of "THE BOOK", the book in which, according to Erdös, God preserves the perfect proofs of mathematical theorems(s. [AZ98]).

The foundational efforts in mathematics towards the simple whole has layed down, respectively, in the valid recognized semantics of mathematics. Thus, the today dominant semantics of set structures makes possible, by the simplicity and generality of the mathematical concept of sets, a standardization of theory-building and therefore an increase of desirable connections (cf. [Bo74]). Since sets can be understood as mathematical abstractions of concepts, mathematics can be effectively integrated by a semantics of set structures into a conceptually developed world. With such an integration into the world, the sense critical judgment of mathematics has to be started.

Concept lattices, described by (esthetically) well drawn line diagrams, deliver a wealth of examples for sense causing integration of mathematical structures in the real world (s. for instance [Wi00b]). This shall be demonstrated here by



Example 3: A music context and its concept lattice

only one small example out of the music area: The inscribed line diagram of Example 3 represents a concept lattice of a formal context, the objects of which are the tonalities of the C-major diatonic scale and the attributes of which are the C-major and -minor triads (a tonality as object has a triad as attribute if this tonality belongs to that triad exactly when the tonality is contained in the triad). The diagram supports the inclusion of the harmonic structure of the C-major diatonicism including its mirror symmetry at the tone d; for instance, one sees (almost with one look) the cyclic-harmonic major-minor-relationships C - a - F - d - G - e - C.

2. The ethics deliver arguments to look about mathematical patterns of thinking and their logical realizations in response to how far they may be chosen sensibly enough in view of the purpose. According to Peirce, ethics win such arguments by "the study of right actions standing in correspondence with purposes which we are willing to accept well-considered ([Pe03a]; p.386). The validity of such argumentative statements can only be secured according to "the rational foundation of the ethics" of Karl Otto Apel [Ap76] by rational argumentation in

the respective community of communication. The thereby postulated "a priori of the community of communication as sense critical condition of possibility and validity" concerns the arguing members in a double manner: "Who namely argues, that one presumes already two things: First a 'real community of communication', which receives each member by a process of socialization, and secondly an 'ideal community of communication' which in principle would be capable to understand adequately the sense of the arguments and to judge definitively their truth" ([Ap76]; p.429). Consequence of such an understood a priori of the community of communication is to strive to the effect that in the real community of communication always a more on the ideal community of communication is realized.

In the real community of communication of mathematicians consists - as already mentioned - a successful strive towards consensus and coherence of greatest possible size. Therefore the most mathematicians see practically no difference between the real and the ideal community of communication and have the opinion that mathematical statements can be principally judged "right" or "false". However, the possibility of yet not recognized paradoxes, which could bring with it certain corrections in the understanding of mathematics, is not impossible according to many mathematicians. Also the value of mathematical concept- and theory-buildings, mathematicians could be judged thoroughly different.

In interdisciplinary communities the judgment of mathematics and its applications are frequently not easy. Since in this case the purposes relate with actions even beyond mathematics, the forms of mathematical thinking have to be functionally examined with respect to their logical realizations. For this, transdisciplinary methodologies are often missing (s. for instance [Wi02b]) and, respectively, the attention about such methodologies.

An important methodology for examining the adequateness of mathematical models, which is unfortunately not much recognized, deliver the representational measurement theory with its representation-, uniqueness-, and meaningfulness theorem (s. first of all [KLSW71]). These measure-theoretical theorems deliver criteria with which the modelling of real vector spaces can be proved as adequate (meaningful). Unfortunately, this proof was performed - as for instance in applications of multivariant statistics (for example: factor analysis) - only seldom so that the inadequate use of mathematical models infrequently become recognized. The reputed statistician Louis Gutman even writes in [Gu77] that in the books on factor analysis after 70 years of research there is not listed even a unique established empirical piece of knowledge (cf. [Wi95]), Excursion 1). That multivariant procedures can be replaced by more adequate, this show for instance the extensive applications of formal concept analysis in the case of the evaluation of repertory tests, with which N. Spangenberg and K. E. Wolff have successfully supported the treatments of anorectic patients [SW93].

3. The logic makes it possible to judge mathematical patterns of thinking and their logical realizations thereupon how far means of representation are available for them to fulfill the given purposes. The formal concepts and concept lattices offer themselves as basic mathematical patterns of thinking because they are

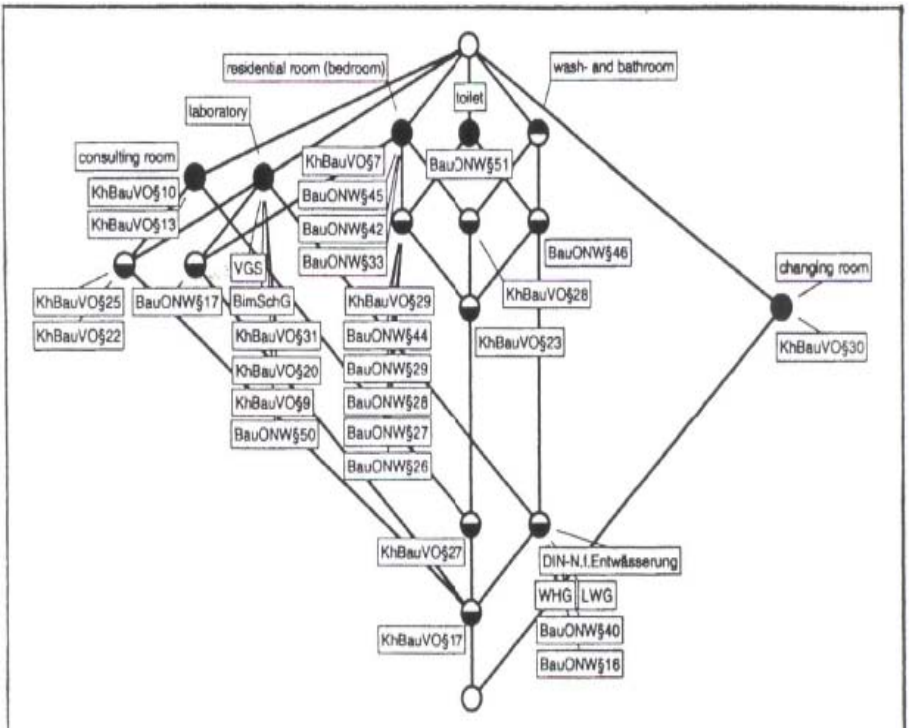
mathematizations of concepts and concept hierarchies, the basic structures of human thinking. Conversely, the concepts and concept hierarchies are understood as logical realizations of their mathematizations.

To make this connection between logic and mathematics more close, the conception of a dyadic mathematics is discussed in [Wi04c] which is based on formal concepts instead of sets. In this way it can - at least initially - be elucidated how mathematical patterns of thinking of a dyadic order theory, contextual mathematical logic, dyadic algebra, and dyadic geometry may be judged in connection with their logical realization as means of legitimate purpose representation. How far this approach can be extended to larger parts of mathematics, this has to be further explored.

At least by an example it should be demonstrated how mathematical patterns of thinking can fulfill a given purpose by logical representations. As example a research project shall be chosen which has been performed by the Darmstadt "research group of concept analysis" in the 90th together with the Department of Building and Housing of the State of "Nordrhein-Westfalen". The task was to develop a prototype of an information system about laws and regulations concerning building constructions which may support first of all architects. The system should be realized with the program TOSCANA [KSVW94] which is based of the theory of formal concept analysis [GW96]. The main purpose of that system was defined to be a support for the planing department and building control office as well as for people that are entitled to present building projects to the office in order to enable these groups to consider the laws and technical regulations in planing, controlling, and implementing building projects (cf. [KSVW94], [EKSW00], [Wi05b]).

The commonly acquired data basis comprised 156 documents about building laws and regulations and furthermore 216 search words (concerning building parts and demands), between those 5.895 relevant relationships were determined in its contens [EKSW00]. As mathematical structure the explorating system takes as a basis a context with 156 formal objects (logically: documents), 216 formal attributes (logically: building parts and demands), and 5.895 object-attribute-pairs (logically: relevant relation between document and building-part/demand). By exploration questions concerning the data basis it should first of all be clarified which documents have to be considered concerning a special building task. To make the answering of such questions possible, a large number of conceptual questioning structures were deduced from the elaborated data context by falling back on the mathematical theory of subcontexts. It was decisive that the concept lattices of the selected subcontexts and - if possible - also their combinations by good readable line diagrams could be represented and therewith could master the main purpose of the exploration system.

Example 4 shows such a concept lattice which is suitable as conceptual questioning structure what is being underlined by the following experience: For testing the readability of the line diagram in Example 4, a secretary was included



Expl. 4: Query structure "functional rooms in a hospital" of a TOSCANA information system about laws and regulations concerning building construction

into the discussion during a working session of the ministry. The secretary became exceedingly surprised when she noticed that in the line diagram the 51 of the building regulations of "Nordrhein-Westfalen" ("BauONW51") stands on the concept with the inscription "toilet" and must therefore be applied only for building toilets. She could not understand that, for instance, "wash- and bathrooms" must not be equipped for handicapped peoples. Also the experts were astonished, but could only determine explicitly, after reading again 51, the relation to the toilet. Only after an extensive discussion, the experts came to the conclusion that by a more general legal point of views also an application of 51 to the "wash- and bathroom" should be demanded. Finally, 51 was also assigned to the "the consulting- and common room", so that 51 became in the diagram the place of the concept with the inscription "KhBauVo27" (cf. [Wi95]). Besides the positive readability, the example makes clear how sometimes first by a process of combinations of mathematical and logical thinking harmonic representations are reached.

Finally, it is still generally to be held that for the sense-critical judgment of mathematics - as also for the normative sciences - Peirce's pragmatism is a fruitful basic idea. The pragmatic maxime, which is central for him, signifies

that the rational content of meaningful thinking consists of the conceivable impressions and effects which may quite probably also have practical consequences [Pe68]. According to this maxime, the determination of sense and meaning of mathematics is genuinely related to its embedding in the real world.

5 Mathematics for Supporting Rational Communication

In the article "communicative rationality, logic, and mathematics" [Wi02c] the following thesis is explained and substantiated:

> *Sense and meaning of mathematics finally lie in the fact that mathematics is able to support the rational communication of humans.*

The self-understanding of mathematics, which is for the theme "presenting mathematics" of central meaning, can be only formed in the community of mathematicians in the frame of controlled rational communication; for this mathematics itself carries decisively the communication about new ideas and results. The far-reaching development of mathematics, which grows out of itself, succeeds first of all so impressive because consent and coherence have the highest priority for mathematicians. This leads to a very restrictive technical language and extremely rigid semantics of mathematics, which facilitates the rational communication about mathematics. Comprehensive thinking in mathematics is supported in particular by mathematical diagrams and pictures.

To understand mathematical thinking as abstraction of logical thinking is basic for the theme "reflecting mathematics", which, first of all, asks about the relationship of mathematics to the real world. Since human thinking is primarily related to the actual realities and activates therefore logical forms of thinking, mathematics support the rational communication of humans if mathematical patterns of thinking can be logically interpreted. The semantics of set structures in mathematics allow the elementary relationship between mathematics and logic that sets are interpretable as extents of concepts. Finally, based on this relationship, a high percentage of established mathematizations concerning the real world can therefore be moved up to the support of rational communication about the real world.

Sense, meaning, and connection of mathematical doing are central categories for the evaluation of integrating mathematics in the real world for the theme "judging mathematics". Since the sense-critical judgment of mathematics relates primarily to the rational purposes, which the mathematized realities shall fulfill according to meaning and connection, mathematics can effectively support the rational understanding about such purposes and about the appertaining realizations. In the sense of the a priori of the community of communication it is even aspired that mathematics can contribute by a suitable transdisciplinary methodology also to interdisciplinary communication, for which suitable representations of logical structures would be provided in an extensive extent.

References

[AZ98] Aigner, M., Ziegler, G.M.: Proofs from THE BOOK. Springer, Berlin (1998)

[Ap76] Apel, K.-O.: Das Apriori der Kommunikationsgesellschaft und die Grundlagen der Ethik. Zum Problem einer rationalen Begründung der Ethik im Zeitalter der Wissenschaft. In: Apel, K.-O. (ed.) Transformation der Philosophie. Bd.2. Suhrkamp-Taschenbuch Wissenschaft 165, Frankfurt, pp. 358–435 (1976)

[Bo74] Bourbaki, N.: Die Architektur der Mathematik. In: Otte, M. (Hrsg.) Mathematiker über die Mathematik, pp. 140–159. Springer, Berlin (1974)

[Da03] Dau, F.: The logic system of concept graphs with negation (and relationship to predicate logic). LNCS (LNAI), vol. 2892. Springer, Heidelberg (2003)

[DK05] Dau, F., Klinger, J.: From formal concept analysis to contextual logic. FB4-Preprint, TU Darmstadt (2005)

[Ek04] Eklund, P. (ed.): ICFCA 2004. LNCS (LNAI), vol. 2961. Springer, Heidelberg (2004)

[EKSW00] Eschenfelder, D., Kollewe, W., Skorsky, M., Wille, R.: Ein Erkundungssystem zum Baurecht: Methoden der Entwicklung eines TOSCANA-Systems. In: [SW00], pp. 254–272

[Fi88] Fischer, R.: Mittel und System: Zur sozialen Relevanz der Mathematik. Zentralblatt für die Didaktik der Mathematik 20(1), 20–28

[GW96] Ganter, B., Wille, R.: Formale Begriffsanalyse: Mathematische Grundlagen. Springer, Heidelberg (1996)

[GW99] Ganter, B., Wille, R.: Contextual attribute logic. In: Tepfenhart, W., Cyre, W. (eds.) ICCS 1999. LNCS (LNAI), vol. 1640, pp. 401–414. Springer, Heidelberg (1999)

[Gu77] Guttman, L.: What is not what in statistics. The Statistician 26, 81–107 (1977)

[Hu92] Husserl, E.: Die Krisis der europäischen Wissenschaften und die tranzendentale Phänomenologie. In: Husserl, E. (ed.) Gesammelte Schriften 8. Felix Meiner Verlag, Hamburg (1992)

[Ke94] Kelle, U.: Empirisch begründete Theoriebildung. Zur Logik und Methodologie interpretativer Sozialforschung. Deutscher Studienverlag, Weinheim (1994)

[KSVW94] Kollewe, W., Skorsky, M., Vogt, F., Wille, R.: TOSCANA - ein Werkzeug zur begrifflichen Analyse und Erkundung von Daten. In: Wille, R., Zickwolff, M. (Hrsg.) Begriffliche Wissensverarbeitung - Grundfragen und Aufgaben, pp. 267–288. B.I.-Wissenschaftsverlag, Mannheim (1994)

[KLSW71] Krantz, D., Luce, R.D., Suppes, P., Tversky, A.: Foundations of measurement, vol. 1, 2, 3. Academic Press, San Diego (1871, 1989, 1990)

[Ma00] Mackensen, K.: Simplizität. Genese und Wandel einer musikästhetischen Kategorie des 18. Jahrhunderts. Bärenreiter, Kassel (2000)

[MW99] Mackensen, K., Wille, U.: Qualitative text analysis supported by conceptual data systems. Quality & Quantity 33, 135–156 (1999)

[MSW99] Mineau, G., Stumme, G., Wille, R.: Conceptual structures represented by conceptual graphs and formal concept analysis. In: Tepfenhart, W., Cyre, W. (eds.) ICCS 1999. LNCS (LNAI), vol. 1640, pp. 423–441. Springer, Heidelberg (1999)

[Oe93] Oehler, K.: Charles Sanders Peirce. Verlag C. H. Beck, München (1993)
[Pe03a] Peirce, C.S.: Aus den Pragmatismus-Vorlesungen. In: Peirce, C.S. (ed.) Schriften zum Pragmatismus und Pragmatizismus. Herausgegeben von K.-O. Apel. Suhrkamp-Taschenbuch Wissenschaft 945. Frankfurt, pp. 337–426 (1991)
[Pe03b] Peirce, C.S.: The three normative sciences. In: Peirce, C.S. (ed.) Edited by the Peirce Edition Project (1893-1913), vol. 2, pp. 196–207. Indiana University Press, Bloomington (1998)
[Pe04] Peirce, C.S.: Eine intellekte Autobiographie. In: Peirce, C.S. (ed.) Semiotische Schriften I. Herausgebenen und übersetzt von Ch. Kösel und H. Pape. Wissenschaftliche Buchgesellschaft, Darmstadt, pp. 64–75 (2000)
[Pe11] Peirce, C.S.: A sketch of logical critics. In: Peirce, C.S. (ed.) Edited by the Peirce Edition Project (1893-1913), vol. 2, pp. 451–462. Indiana University Press, Bloomington (1998)
[Pe68] Peirce, C.S.: Über die Klarheit unserer Gedanken. Einleitung, Übersetzung, Kommentar von K. Oehler. Klostermann, Frankfurt (1968)
[Pe92] Peirce, C.S.: Reasoning and the logic of thinks. In: Ketner, K.L. (ed.) with an introduction by Ketner, K.L., Putnam, H. Havard Univ. Press, Cambridge (1992)
[Pe93] Peirce, C.S.: Konventionen und Regeln der Existentiellen Graphen. In: Peirce, C.S. (ed.) Phänomen und Logik der Zeichen. Herausgegeben und übersetzt von H. Pape. Suhrkamp-Taschenbuch Wissenschaft 425, 2, Aufl., Frankfurt, pp. 139–155 (1993)
[Pr98] Prediger, S.: Kontextuelle Urteilslogik mit Begriffsgraphen. Ein Beitrag zur Restrukturierung der mathematischen Logik. Dissertation, TU Darmstadt. Shaker Verlag, Aachen (1998)
[Pr01] Prediger, S.: Mathematik als kulturelles Produkt menschlicher Denktätigkeit und ihr Bezug zum Individium. In: Lengnink, K., Prediger, S., Siebel, F. (Hrsg.) Mathematik und Mensch: Sichtweisen der Allgemeinen Mathematik, pp. 21–36. Verlag Allgemeine Wissenschaft, Mühltal (2001)
[Se01] Seiler, T.B.: Begreifen und Verstehen. Ein Buch über Begriffe und Bedeutungen. Verlag Allgemeine Wissenschaft, Mühltal (2001)
[SW93] Spangenberg, N., Wolff, K.E.: Datenreduktion durch Formale Begriffsanalyse von Repertory Grids. In: Scheer, J.W., Catina, A. (Hrsg.) Einführung in die Repertory Grid-Technik. Bd.2: Klinische Forschung und Praxis, Bern, pp. 38–54 (1993)
[SWW01] Strahringer, S., Wille, R., Wille, U.: Mathematical support for empirical theory building. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 169–186. Springer, Heidelberg (2001)
[SW93] Spangenberg, N., Wolff, K.E.: Datenreduktion durch Formale Begriffsanalyse von Repertory Grids. In: Scheer, J.W., Catina, A. (Hrsg.) Einführung in die Repertory Grid-Technik. Bd.2: Klinische Forschung und Praxis, Bern, pp. 38–52 (1993)
[SC96] Strauss, A., Corbin, J.: Grounded Theory: Grundlagen qualitativer Sozialforschung. Beltz, Weinheim (1996)
[SW00] Stumme, G., Wille, R. (Hrsg.): Begriffliche Wissensverarbeitung: Methoden und Anwendungen. Springer, Heidelberg (2000)
[Th93] Thamm, S.: Kurs "Formale Begriffsanalyse" - Eine Brücke zwischen formalem und inhaltlichem Denken. Staatsexamensarbeit, TH Damstadt (1993)

[Wi82] Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered sets, pp. 445–470. Reidel, Dordrecht (1982)

[Wi86] Wille, R.: Mathematik für Sozialwissenschaftler. Vorlesungsskript, TH Darmstadt 1986/87

[Wi87] Wille, R.: Bedeutungen von Begriffsverbänden. In: Ganter, B., Wille, R., Wolff, K.E. (Hrsg.) Beiträge zur Begriffsanalyse, pp. 161–211. B.I.-Wissenschaftsverlag, Mannheim (1887)

[Wi95] Wille, R.: Begriffsdenken: Von der griechischen Philosophie bis zur Künstlichen Intelligenz heute. In: Diltheykastanie, Ludwig-Georgs-Gymnasium, Darmstadt, pp. 77–109 (1995)

[Wi96] Wille, R.: Restructuring mathematical logic: an approach based on Peirce's pragmatism. In: Ursini, A., Agliano, P. (eds.) Logic and Algebra, pp. 267–281. Marcel Dekker, New York (1996)

[Wi97] Wille, R.: Concept Graphs and Formal Concept Analysis. In: Lukose, D., Delugach, H., Keeler, M., Searle, L., Sowa, J. (eds.) ICCS 1997. LNCS (LNAI), vol. 1257, pp. 290–303. Springer, Heidelberg (1997)

[Wi00a] Wille, R.: Contextual logic summary. In: Stumme, G. (ed.) Working with conceptual structures: Contributions to ICCS 2000, pp. 265–276. Shaker-Verlag, Aachen (2000)

[Wi00b] Wille, R.: Begriffliche Wissensverarbeitung: Theorie und Praxis. Informatik Spektrum 23, 357–369 (2000)

[Wi01a] Wille, R.: Allgemeine Mathematik - Mathematik für die Allgemeinheit. In: Lengnink, K., Prediger, S., Siebel, F. (Hrsg.) Mathematik und Mensch: Sichtweisen einer Allgemeinen Mathematk, pp. 3–19. Verlag Allgemeine Wissenschaft, Mühltal (2001)

[Wi01b] Wille, R.: Lebenswelt und Mathematik. In: Hauskeller, C., Liebert, W., Ludwig, H. (Hrsg.) Wissenschaft verantworten: soziale und ethische Orientierung in der Technischen Zivilisation, pp. 51–68. Agenda-Verlag, Münster (2001)

[Wi01c] Wille, R.: Mensch und Mathematik: Logisches und mathematisches Denken. In: Lengnink, K., Prediger, S., Siebel, F. (Hrsg.) Mathematik und Mensch: Sichtweisen einer Allgemeinen Mathematk, pp. 139–158. Verlag Allgemeine Wissenschaft, Mühltal (2001)

[Wi02a] Wille, R.: Begriffliche Wissensverarbeitung in der Wirtschaft. Information - Wissenschaft und Praxis (Organ der Deutschen Gesellschaft für Informationswissenschaft und Informatiospraxis e.V.), vol. 53, pp. 149–160 (2002)

[Wi02b] Wille, R.: Transdisziplinarität und Allgemeine Wissenschaft. In: Krebs, H., Gehrlein, U., Pfeifer, J., Schmidt, J.C. (Hrsg.) Perspektiven interdisziplinärer Technikforschung: Konzepte, Analysen, Erfahrungen, pp. 73–84. Agenda-Verlag, Münster (2002)

[Wi02c] Wille, R.: Kommunikative Rationalität, Logik und Mathematik. Mathematische Semesterberichte 49, 167–183 (2002)

[Wi03] Wille, R.: Conceptual content as information - basics for conceptual judgment logic. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS (LNAI), vol. 2746, pp. 1–15. Springer, Heidelberg (2003)

[Wi04a] Wille, R.: Implicational concept graphs. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 52–61. Springer, Heidelberg (2004)

[Wi04b] Wille, R.: Sind unsere Vorstellungen von Raum und Zeit richtig? Oder: Besteht ein Kontinuum aus Punkten? In: Hefendehl-Hebeker, L., Hussmann, S. (eds.) Mathematikdidaktik: Zwischen Fachorientierung und Empirie, pp. 266–279. Franzbecker Verlag, Hildesheim (2003)

[Wi04c] Wille, R.: Dyadic mathematics - abstractions from logical thought. In: Denecke, K., Erné, M., Wismath, S.L. (eds.) Galois connections and applications, pp. 453–498. Kluwer, Dordrecht (2004)

[Wi05a] Wille, R.: Formal Concept Analysis as mathematical theory of concepts and concept hierarchies. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 1–33. Springer, Heidelberg (2005)

[Wi05b] Wille, R.: Mathematik - präsentieren, reflektieren, beurteilen. In: Lengnink, K., Siebel, F. (Hrsg.) Mathematik - präsentieren, reflektieren, beurteilen, pp. 3–19. Verlag Allgemeine Wissenschaft, Mühltal (2005)

The Role of Concept, Context, and Component for Dependable Software Development*

Vasu Alagar[1], Mubarak Mohammad[1], and Kaiyu Wan[2]

[1] Concordia University, Montreal, Canada
{alagar,ms_moham}@cse.concordia.ca
[2] East China Normal University, Shanghai, PRC
kywan@cs.ecu.edu.cn

Abstract. Software that impact our lives are embedded in the environment in which we act and hence our security and safety are dependent on its flawless functioning. An assessment of dependability of such embedded software systems includes an assessment of the process to develop the system and the system's observable properties. Dependability criteria depend on the domain in which the software system is to serve. Thus, it should be formulated from domain concepts. Concepts in the domain should be analyzed to construct components. A component of the system may function as expected in one context of application and may fail to function as expected in another context. The system is dependable if the services resulting from every interaction between system components satisfy the dependability criteria in every context of operation. This paper explores the roles of concept analysis and context in determining dependability criteria at the domain level, the role of domain models in an automatic derivation of components and component-based systems, and the integrated role of context and components in the construction of context-aware and service-oriented systems.

1 Introduction

A software system is *dependable* if it delivers the results for which it was designed and no adverse effects are felt in the environment during and after the delivery of the results. Software that impact our lives are embedded in the environment in which we act and hence its behavior should remain dependable. It is the goal of this paper to craft the role of *concept* and *context* in formulating a dependability criterion, the relationship between concept and a software *component* and the role of a context-aware *component-based development* (CBD) in a convincing assessment of dependability in the deployed software.

In literature the terms *dependability* and *trustworthiness* are used interchangeably. Without trusting the behavior of a system it cannot be depended upon. Conversely, a system is regarded as dependable only if it is trustworthy. Let us review some examples of software systems that we depend upon in our lives. Embedded software that drive smart medical devices and on-line health care systems are prime examples. They

* This research is supported by a Research Grant from Natural Sciences and Engineering Research Council of Canada.(NSERC)

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 34–50, 2010.

bring enormous benefits, but when they malfunction or fail the patients are in peril. Safety, security, and privacy are the attributes that determine trust in such applications. In the sector of on-line finance and E-commerce, the system can be trusted only if it is secure, provides timely service, enforces obligations, and affords privacy. In transportation domain, aircrafts have autopilots installed in them, which once initialized will take away the control of a pilot, resulting in cruising pleasure and occasionally resulting in a severe accident. Safety, security, reliability, and availability are all essential attributes to ensure trust in such systems. In the energy sector, large power grids and nuclear power plants should be safe-guarded and protected without fail. Without a direct evidence of safety and security software that monitors and manages these systems cannot be trusted. These examples illustrate the types and severity of risks that vary from one sector (domain) to another sector. So the first lesson we learn is that dependability criteria is to be composed from attributes that primarily include *safety*, *security*, *reliability*, and *availability*. Since attributes are related to concepts and in turn concepts belong to a domain of application we learn the second lesson that the dependability criterion is domain-dependent and should be formulated from domain concepts. What is deemed as dependable for health care domain may not be relevant to transportation domain, and what is accepted as dependable for ground transportation may not be acceptable as sufficiently dependable for air transportation.

Once the application domain of a large complex system is prescribed, dependability is to be *articulated* at the *requirements* level and *evaluated* at the *entire system* level. It is natural therefore to include the essential critical properties of the domain in formulating the dependability criteria. It may be theoretically infeasible to know all of the properties that are critical. Making an exhaustive list of such properties may pose a problem for the system developer in the assessment of those properties in the system. To be practical, the dependability property must involve only the essential claims and the formulated dependability property must be formally verified in the system. Formal verification is a daunting task. If there are risks, notwithstanding the verified properties, they must be made explicit to users of the system in order that clients may understand it and determine how well they want to trust the system. In particular, for software in privacy and safety-critical domains the dependability argument must be in the form of direct evidence that is verifiable and can be audited by a third party who needs not be an expert.

Software is but one part of the entire system. A component of the software may function as expected in one context of application and may fail to function as expected in another context [7, 10]. So, it is essential to integrate with software components the contexts in which they are to be used. Towards that, a software system is viewed as the sum of three parts, the software unit S with which the client will have a direct interaction, its embedding E_i (its development framework and third party components), and the environment E_o in which it will be deployed. An application within a domain has a specific context of application. The features of E_i will lead to determining those *internal contexts* which impact the generation of services of the components in S. The features of E_o will prescribe the *external contexts* in which the services of S are to be provided. The external contexts also form the basis of *context-awareness* of the system. The dependability attributes must be determined in that context, the closure of the internal and external contexts, and a dependability case should be formulated. This argument should

be an expression ϕ that includes the global context information, the critical properties of the software, assumptions on its embedding, its environmental constraints, and a direct evidence that the specified properties are satisfied. This dependability argument will have to be resolved by a rigorous proof that $M \models \phi$, where M is the model composed of S, E_i and E_o.

During system development external contexts may not change, however some new internal contexts may be generated which in conjunction with other internal contexts may modify the set of internal contexts. The essence of dependable development is to ensure that the property ϕ remains invariant through the development cycle. Thus the initial model M undergoes several *refinements*, say $M_1, M_2, \ldots, M_k$, where $M = M_1$ and M_i is a refinement of M_{i-1}. Typically a refinement is a conservative extension of structure and behavior, adding more details while conservatively extending the original behavior. Let $\phi_1 = \phi$. For refinement M_2 we need to construct an evidence ϕ_2 such that $M_2 \models \phi_2$ and prove $\phi_2 \Rightarrow \phi_1$. Then it follows that $M_2 \models \phi_1 (= \phi)$. That is, the dependability criterion is satisfied by the model M_2. In general, for successive refinements $M_2, \ldots, M_k$, of $M(= M_1)$ we need to construct evidences ϕ_i, $2 \leq i \leq k$ such that $M_i \models \phi_i$, $2 \leq i \leq k$ and provide a proof of $\phi_i \Rightarrow \phi_{i-1}$. With this proof we can claim that the implemented system M_k satisfies the dependability criterion formulated at the domain level. The proofs of $\phi_i \Rightarrow \phi_{i-1}$, $i = 2, \ldots, k$ form the chain of *evidence* that the system is constructed without losing the stated dependability property. The construction of this chain of evidence is the most challenging problem in the development of dependable systems. We believe that understanding the roles of component, context, and concept and skilfully exploiting their triangular web of interactions we may break the complexity barrier in the development of dependable systems. We sketch these details in the rest of the paper.

2 Basic Concepts - Concept, Context, Component

According to OED [15] the term "concept" means an *abstract idea* and is derived from the Latin word *conceptum* meaning "something conceived". Traditional philosophical view has lead to view concept as a pair (*extent, intent*), where the extent consists of all objects *belonging* to the concept and intent is the set of all attributes *shared* by the objects. Because of the inherent difficulty in determining all objects that belong to a concept and identifying all the attributes shared by them, concept classification and analysis takes place within a specific context. But the notion of context itself is not defined. Formal Concept Analysis (FCA) [8] provides a framework for structuring, analyzing, and visualizing concepts and concept hierarchies. For further discussion it is sufficient that we agree that FCA requires a priori knowledge of context and a set of objects of some relevance in that context.

According to OED the term "context" means the circumstances that form the setting for an event, statement, or idea. A circumstance is a condition involving, in general, different types of entities. As an example, the setting for a "seminar event" is a condition involving entities *speaker*, *topic*, *time*, *location*. When each entity is assigned a value from the domain associated with that entity, and if the condition is met then the seminar is to be held. A common social usage of the word context is illustrated in OED

by the quotation "I wish honorable gentlemen would have the fairness to give the entire context of what I did say, and not pick out detached words.(Cobden, Speeches 46, 1849, quoted in the OED)". Although the word context has been used for a long time in many scientific descriptions, literary essays, and in philosophical discourses, its meaning was always left to the reader's understanding. In one of the earlier papers, Clark and Carlson [5] state that

> Context has become a favorite word in the vocabulary of cognitive psychologists and that it has appeared in the titles of a vast number of articles. They then complain that the denotation of the word has become murkier as its uses have been extended in many directions and deliver the now widespread opinion that context has become some sort of "conceptual garbage can".

Context is studied as a formal object for logical reasoning in Artificial Intelligence. Intensional Logic [6,18], a family of mathematical formal systems that permits expressions whose value depends on *hidden context*, came into being from research in natural language understanding. According to Carnap [3], the real meaning of a natural language expression whose truth-value depends on the context in which it is uttered is its *intension*. The *extension* of that expression is its actual truth-value in the different possible contexts of utterance, where this expression can be evaluated. Basically, intensional logics add *dimensions* to logical expressions, and non-intensional logics can be viewed as *constant* in all possible dimensions, i.e. their valuation does not vary according to their context of utterance. *Intensional operators* are defined to *navigate* in the context space. In order to navigate, some dimension and *tags* (or indexes) are required to provide placeholders along dimensions. These dimension tags, along with the dimension names they belong to, are used to define the context for evaluating intensional expressions.

Based upon an intuitive understanding, context was freely used to describe different computing scenarios. However, when system developers were faced with the development of large complex systems which must be context-aware it was necessary to invent some form of working definition of context. Context sensing, context representation, contextual analysis, and contextual interpretation are necessary steps in developing pervasive computing applications, sensory networks and mobile computing, Human-Computer Interaction (HCI), semantic web, knowledge-based systems, and analysis of computer programs. Without some working definition of context such systems can not be designed. This necessity has lead to many ad hoc but useful technical working definitions of the notion of context.

Following the intensional logic paradigm in which dimensions and tags were used to describe contexts, Wan [19] formalized context as a *typed relation*, introduced a syntax for it, and provided a set of operators. Based on it a context calculus was developed and a context toolkit has been implemented. This toolkit can be used as a "plug-in" for the applications cited above.

Component is a software engineering concept. The notion of component provides a means to model an entity which provides services to its environment and may require services from its environment. The two types of services are *provided* services and *required* services. Provided services are the results of operations performed within a component. Required services are the prerequisite services that the component needs

in order to provide its services. The notion of interface provides a means to specify the services that are provided and required by the component.

A service must fulfill the service contract. The notion of contract provides a powerful mechanism to specify and enforce safety, security, reliability, and availability requirements. At design time, contract specification is the basis for a formal validation of dependability requirements to ensure that there are no contradicting dependability requirements, and a formal verification that the design of component respects the dependability requirements. At run-time, contract implementation enforces the dependability requirements in the behavior of a component. In order to provide an evidence of dependability the implemented services should satisfy the following properties:

- safety: services do not cause danger to the environment.
- security: there is no unauthorized access or improper alteration to information.
- reliability: correct services are provided despite any failure.
- availability: services are provided with no interruption.

If the services provided by a component satisfy all the above properties at different contexts of invoking the component we say the component is dependable. A system is composed with many components that interact among themselves. If the services resulting from every interaction satisfy the above properties in every context of system's operation we say the system is dependable. Therefore, it is necessary to construct contexts and integrate them with components in the system, and the dependability criterion ϕ is formulated at the requirements level to capture safety, security, reliability, and availability requirements. Since the requirements originate from the application domain, concepts in the domain should be analyzed to construct components that are dependable at different contexts. This is the context of our current research.

3 Concepts in Contexts

In this section we give a brief outline of the context formalism from Wan [19] and give a logical contextual reasoning on concept classes, viewed as the knowledge referenced by contexts.

3.1 Context Definition

Several notable works on context formalization have been surveyed in [1]. With neither an abstract nor a concrete representation of context, these papers discuss different kinds of logic for reasoning with context. Motivated by system and language requirements Wan [19] modeled context as a typed relation. This formalization retains the first class status of contexts and logical reasoning ability, the twin principles of McCarthy [11]. The advantage of Wan's formalism is that contexts have a representation (structure) and a semantics based on the knowledge enveloped by the context. Concepts defined within a specific context, as in FCA, can be viewed as a world of knowledge whose pointer is the specific context.

In order to define a context Wan [19] started with a set of dimensions and their associated types. Context is a multi-dimensional object, as can be inferred from a computing perspective. Naturally dimensions and their types for contexts arise from the

Table 1. Context Operators and their Precedence

operator name	symbol	meaning
Projection	$\downarrow$	Domain Restriction
Hiding	$\uparrow$	Range Restriction
Union	$\sqcup$	Set Union
Intersection	$\sqcap$	Set Intersection
Difference	$\ominus$	Set Difference
Override	$\oplus$	Function overwrite
Undirected Range	$\rightleftharpoons$	Range of simple contexts with same domain
Directed Range	$\rightharpoonup$	Range of simple contexts with same domain

domain of interest. Therefore we define a typing function $\tau : DIM \rightarrow \mathcal{T}$, which associates with every dimension in the set $DIM = \{X_1, X_2, \ldots, X_n\}$ a type from the set $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$. Let $\tau(X_i) = T_i$, $T_i \in \mathcal{T}$. Define a context c as an aggregation of ordered pairs (X_j, v_j), where $X_j \in DIM$, and $v_j \in \tau(X_j)$. In principle, DIM may be infinite. We deal only with finite contexts. So, it is sufficient to let DIM be finite. We use the notation $[X_1 : v_1, \ldots, X_k : v_k]$, instead of the strict mathematical notation $\{(X_1 \mapsto v_1), \ldots, (X_k \mapsto v_k)\}$ for contexts. For a context $c = [X_1 : v_1, \ldots, X_k : v_k]$, we also write $(X_i, v_i) \in c$, for $i = 1, \ldots, k$. In the rest of the paper by context we mean a finite non-empty context in which all dimensions are distinct, and omit explicit reference to τ unless our discussion demands it.

Context Modification. Context operators are defined on a set of contexts of a specific type τ. These operations provide a means for constructing and modifying contexts dynamically. The context operators include standard binary set operators that take contexts as operands and return a context. An essential set of operators is given in Table 1. In the table, the operators within a rectangular box have equal precedence while the rectangular boxes are arranged in decreasing order of precedence. Formal definitions are given in Wan [19].

3.2 Two Algebraic Structures

Following the notation of McCarthy [11] we write $\mathbf{ist}(c, P)$ to mean that P is true in context c. We write $W(c)$ to denote the set of propositions that are true in context c. That is, $W(c) = \{P \mid P \in \mathcal{P} \wedge \mathbf{ist}(c, P)\}$. The set $W^*(c)$ denotes the closure of $W(c)$. We define two kinds of partial orders on the set S of contexts.

Generality of First Kind. Suppose for contexts $c_1, c_2 \in S$, the world $W(c_2)$ referenced by c_2 is a restricted subset of the information in the world $W(c_1)$ referenced by c_1. The restriction is often imposed by constraints on the entity described in the world $W(c_1)$. Then the information in the world $W(c_2)$ is more precise or special than the information $W(c_1)$. This kind of sub-setting can be realized either semantically or syntactically. In the former case, a logical expression may be used. An example is $W(c_1)$ is the set of students and $W(c_2)$ is the set of married students. In the case of syntactic sub-setting, more dimensions are added to the set $dim(c_1)$ such that the world referenced by new

contexts obtained by the addition of dimensions do not go outside $W(c_1)$. Formally a context c_2 is constructed such that $dim(c_2) = dim(c_1) \cup D$, where $D \subset DIM$, $D \cap dim(c_1) = \emptyset$, such that $W(c_2) \subset W(c_1)$. Thus, the additional dimensions in the set $dim(c_2) \backslash dim(c_1)$ have no extraneous influence on the information along the dimensions in the set $dim(c_1)$, instead it can only constrain it. An example is the context $c_2 = [GPS : Newyork, TIME : 10, NS : 12, EW : 3]$ with $c_1 = [GPS : Newyork, TIME : 10]$, with the interpretation that every event happening in context c_2 can be seen from context c_1. Hence, if P is a valid formula in $W^*(c_2)$ then it is possible to prove in context c_1 that P is true in context c_2. We define such a relationship between contexts as *visible*. Formally, a context $c_2 \in S$ is said to be *visible* from context $c_1 \in S$, written $c_1 \succeq c_2$, if $c_1 \sqsubset c_2$ and $W^*(c_2) \subset W^*(c_1)$. Hence it follows that $\mathbf{ist}(c_2, P) \Rightarrow \mathbf{ist}(c_1, \mathbf{ist}(c_2, P))$ if $c_1 \succeq c_2$. The relation $\succeq$ is a reflexive partial order on the set S. For the poset $(S, \succeq)$ maximal, minimal, greatest, and least elements are defined in the usual manner.

Generality of Second Kind Let $c_1, c \in S$, $dim(c_1) \subset dim(c)$. Suppose the world $W(c_1)$ referenced by c_1 contains information about one entity and the world $W(c_2)$ referenced by any context c_2, $dim(c_2) \subseteq dim(c) \setminus dim(c_1)$, contains information about another entity not related to the first one. Then we can conclude that

- the context $c = c_1 \sqcup c_2$ references the world $W(c) = W(c_1) \cup W(c_2)$, where $W(c_1) \cap W(c_2) = \emptyset$, $W^*(c_1) \subset W^*(c)$, and $W^*(c_2) \subset W^*(c)$,
- every formula P_1 that is valid in c_1 is valid in c, and
- every formula P_2 that is valid in c_2 is valid in c.

Moreover we can also conclude that if a formula P is valid in c then P is valid in either c_1 or in c_2. It may not be valid in both. We say context c is more general than c_1 and write $c \gtrsim c_1$. It is easy to see that $c \gtrsim c_2$. From the discussion above it follows that if $c \gtrsim c_1$, and $c \gtrsim c_2$ then $\mathbf{ist}(c, P) \Rightarrow \mathbf{ist}(c_1, P) \wedge \mathbf{ist}(c_2, P))$. The relation $\gtrsim$ is a irreflexive partial order on the set S. For the poset $(S, \gtrsim)$ maximal, minimal, greatest, and least elements are defined in the usual manner.

3.3 Reasoning

We let the worlds referenced by contexts to be formalized in propositional logic and give rules for reasoning within a context. For predicate logic worlds we have developed a set of rules, not included in this paper.

Distributes on Logical Connectives

$$\circ \in \{\wedge, \vee, \rightarrow\} : \qquad \frac{\mathbf{ist}(c, P) \circ \mathbf{ist}(c, Q)}{\mathbf{ist}(c, P \circ Q)} \tag{1}$$

$$\circ \in \{\wedge, \rightarrow\} : \qquad \frac{\mathbf{ist}(c, P \circ Q)}{\mathbf{ist}(c, P) \circ \mathbf{ist}(c, Q)} \tag{2}$$

$$\frac{\mathbf{ist}(c, \mathbf{ist}(c', P) \vee Q)}{\mathbf{ist}(c, \mathbf{ist}(c', P)) \vee \mathbf{ist}(c, Q)} \tag{3}$$

Modes Ponen : Logical inference:

$$\frac{\mathbf{ist}(c, P \rightarrow Q), \mathbf{ist}(c, P)}{\mathbf{ist}(c, Q)} \tag{4}$$

The following two *lifting rules* allow a valid formula in one context to become valid in another context.

Enter Entering from a context c':

$$\frac{\mathbf{ist}(c', \mathbf{ist}(c, P))}{\mathbf{ist}(c, P)} \tag{5}$$

Exit Exiting from a context c:

$$\frac{\mathbf{ist}(c, P), c' \succeq c}{\mathbf{ist}(c', \mathbf{ist}(c, P))} \tag{6}$$

Formal System. With the set $S(\tau)$ of simple contexts (contexts with distinct dimensions) we associate a tuple $W(\tau) = \{\langle L(\tau), F(\tau), I(\tau)\rangle\}$, where $F(\tau)$ is a set of facts and $I(\tau)$ is a set of inference rules, both expressed in the language $L(\tau)$. Assume that $\mathcal{P}$ is a set of propositions and $\mathcal{P} \subset L(\tau)$. The transitive closure of all formulas derived from $F(\tau)$ is the world $W^*(\tau)$ describable by the contexts in $S(\tau)$. $W(\tau)$ is a formal system.

4 Concepts in Components

In this section we discuss the construction of dependability criteria and the derivation of components from domain concepts.

4.1 Formulating Dependability Criteria

According to Jackson et al.,[10] "it might make sense to demand dependability from a car in its entirety, it makes less sense to demand the same of a large software system". Yet, for many models of modern day cars that are equipped with thousands of embedded systems proving dependability is not an easy task. However, we can at least define dependability in terms of safety, security, reliability, and availability. For a software system of large complex application it is much harder to define a metric that can be a measure of its dependability. A modest goal should be to include particular properties in the dependability criteria and include rigorous methods for assessing the dependability criteria during software system development. Regardless of the size and complexity of the system, the properties to be included in dependability criteria are independent of any development method and hence must be determined at the domain level. Contexts that are pertinent to the concepts and their interpretations should also be constructed at the domain level. Failing to include context may lead to unsafe systems. Example 1 illustrates this point.

Example 1. *This example is taken from [1], and perhaps the starting point for a formal treatment of context by McCarthy [11]. MYCIN [16] was introduced as a computer-based medical consultation system in 1976. Among the many types of users of MYCIN, physicians could use it on treating bacterial infections of the blood. When MYCIN was given the information "the patient has* Chlorae Vibrio*" it recommended two weeks of* tetracycline *treatment and nothing else. What it did not reveal was that there is massive* dehydration *during the course of the treatment. While the administration of* tetracycline *would cure the bacteria, the patient would perish long before that due to diarrhea.*

The moral is that context must be included with services. We discuss the design aspects of this integration in Section 4. An important aspect of domain analysis should be to identify and represent relationship between concepts. Failing to do so will lead to unsafe software systems. Here is an example taken from [10].

Example 2. *Emergency care units may have a dozen or more different medical devices connected to the same patient. These devices are designed and developed in isolation, but they form an accidental system (that is, a system constructed without conscious intent) whose components interact through the patients physiology and through the cognitive and organizational faculties of the attending physicians and nurses. Each device typically attempts to monitor and support the stabilization of some parameter (heart rate, breathing, blood chemistry) but it does so in ignorance of the others even though these parameters are physiologically coupled. The result can be suboptimal whole-body stabilization and legitimate concern that faults in a device, or in its operation, may propagate to other devices. Because they are designed in isolation, the devices have separate operator interfaces and may present similar information in different ways and require similar operations to be performed in different ways, thereby inviting operator errors. A consequence of accidental system construction is that components may come to be used in contexts for which they were not designed and in which properties (typically internal failures and response to external faults) that were benign in their original context become more serious.*

In summary, the dependability criteria construction requires the following activities.

1. Identification of *critical* concepts for inclusion in the dependability criteria.
2. Formalization of the relationship between critical concepts and their attributes.
3. Analysis of included concepts to remove redundancy and contradictions.
4. Formulation of the criteria as a formal expression.

4.2 A Formal Model of Dependable Components and Systems

We formally describe components and the systems composed from them in a language, called *Trustworthy Architecture Description Language* (TADL) [12,13]. In TADL, a component definition includes functional, structural, and non-functional (trustworthiness) specifications. Figure 2 depicts the structure of TADL for defining trustworthy component and component-based systems. The main building blocks of TADL structure are given below.

- *Component definition:* A component provides and requests services through public interfaces. Also, it defines attributes that define local value-type properties.
- *Architecture definition:* A component can be primitive or composite. The components of a composite component are connected using connectors. An architecture specifies a set of connectors along with attachment specifications for the connectors. It also defines structural constraints. A component can have multiple possible architectures. A component-based system is a composition of components with a predefined architecture or set of possible architectures.
- *Safety contract:* A contract defines the safety requirements that govern the interactions that occur at the interfaces of a component. Also, it defines time constraints that regulate the service requests and responses so that the reactions of a component respect any timeliness requirements. Predictability specification ensures that component reactions are precisely defined. For every service request there should be at least one defined response.
- *Security mechanism:* The security mechanism is based on role-based security access control. The mechanism restricts access of services and data parameters to authorized users only. Security policies are defined and associated with service definitions. A service request is executed only if its corresponding security policy is satisfied. Also, a service response is executed only if its corresponding security policy is satisfied.
- *Reliability and availability:* The definitions of reliability and availability are based on frequency and duration of service failures and repair durations. A failure is a deviation from the correct service behavior. It is indicated by a violation to the functional or non-functional requirements including those of safety and security. A repair is a change from the state of service failure to the state of correct service. Based on the frequency and severity of service failures the acceptable level of reliability is defined. Based on the duration of service failure time the acceptable level of availability is defined. The component implementation and maintenance should guarantee the repair time. The failures, repairs, and the acceptable levels of reliability and availability are formally defined in the component contract.

In order to verify that components satisfy a contract we need to specify the behavior of each component as well as their combined behavior. We model the behavior as an extended timed automata. The timed automata specification includes safety properties, security policies, and failure and repair specification [13]. This enables us to use a formal model checking procedure to verify safety, security, reliability, and availability in one unified approach. We use UPPAAL [2] model checker. We provide an automatic approach for generating component behavior using a model transformation technique.

4.3 Transforming Concepts to Components

A domain is a set of applications that share similar requirements, capabilities, and data. Domain engineering is the set of activities that define, model, construct and catalogue a set of artifacts specific to the domain. The artifacts include a model, architectures, components, applications, contexts of operations, and dependability criteria. Domain engineering is an important first step in developing software systems. At the core of domain

engineering, *domain analysis* is used to capture and classify the domain knowledge. It identifies the requirements that are common for all products in the domain as well as the requirements that are specific to each product. The collected requirements must include the required functionality, the context of operation for each functionality, and the dependability criteria that must be satisfied by the operations. For example, the domain of automotive industry deals with designing, manufacturing, and marketing motor vehicles. A car, for example, contains many control systems such as *cruise control*, *cooling and heating*, *stability control*, *anti-lock braking*, and *fingerprint-based security systems*. Domain analysis provides an understanding of each system, its interactions with other systems, the constituent components in a system, their functional and dependability requirements, the context of operation for each function, and the data and events stored and communicated between them. The results of the domain analysis is a *domain model* which consists of knowledge about the domain and all its applications and its reusable components. This knowledge can be stored in a knowledge base which contains vocabulary of the domain anatomy. This knowledge forms the foundation based on which software systems are developed.

From the domain model, a *domain architecture* is developed to form the basis for all domain products. The architecture is further refined to define the constituent *reusable components*. *Domain applications* are designed based on the domain architecture and developed by reusing existing domain components. Thus, domain analysis plays a key role in developing dependable systems. As a result, finding an effective method for domain analysis becomes a necessary task for building dependable systems. Although the importance of domain analysis was recognized in the literature [14, 4] no formal method was put forth for domain analysis for constructing dependable component-based systems. FCA may have the potential to provide a formal basis for domain analysis and domain modeling.

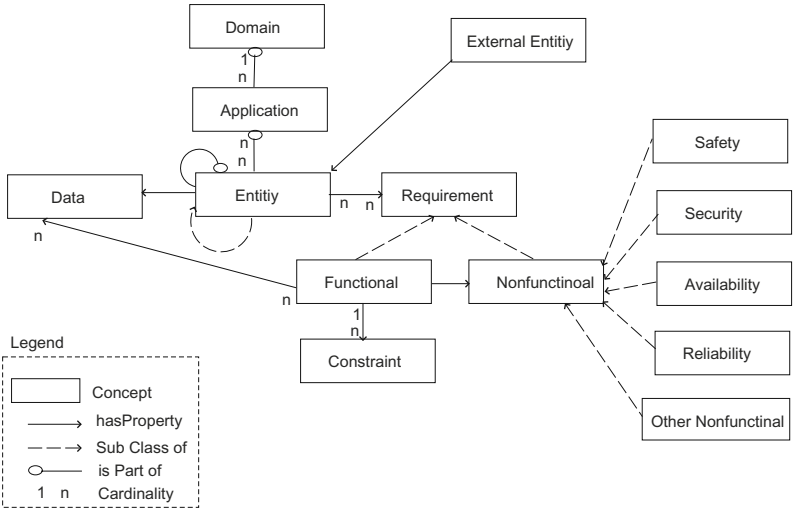


Fig. 1. An ontology for domain analysis

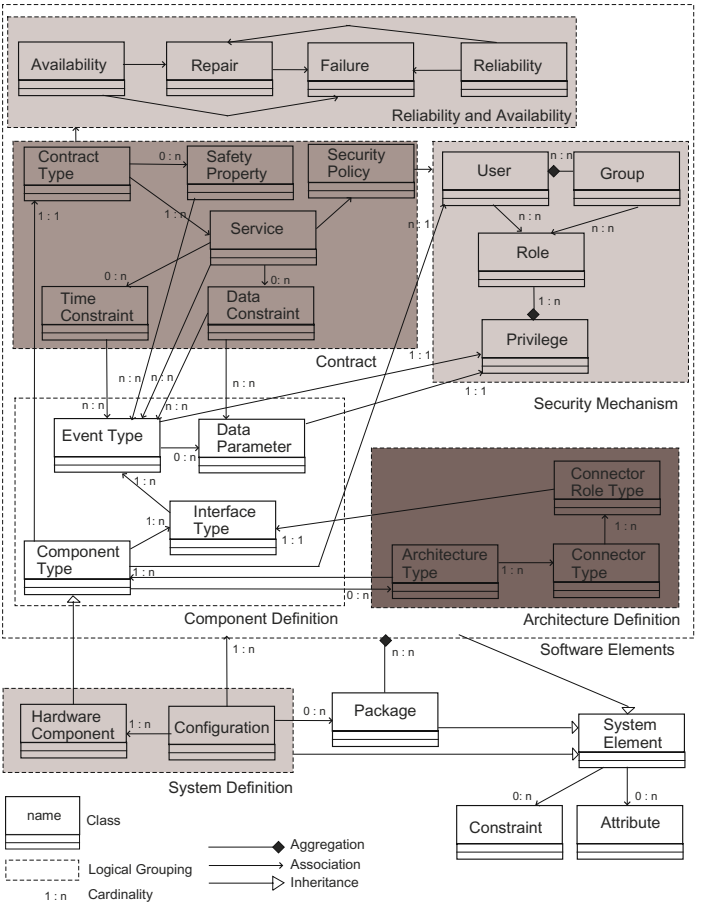


Fig. 2. TADL Structure

During domain analysis, FCA techniques are used to extract domain objects and their attributes. Then, formal contexts are built, where each formal context is a triple (G, M, I) such that G is the set of objects, M is the set of attributes, and $I \subseteq G \times M$ is a binary relation. Formal contexts are used to build concept lattices which contain the formal concepts of the domain. Then, the *reduced labeling* [8] technique is used to reduce the lattice. Finally, formal concepts are extracted from the lattice to define ontology concepts.

Mohammad [13] designed an ontology, shown in Figure 1, for representing the knowledge captured during domain analysis. Using the tool *Protege* [17] an analysis of the ontology is done. When creating the ontology, using the tool the OWL [9], language specification is automatically created. This specification is an XML file which is then transformed to a component architecture in the XML version of TADL, the language that describes the architecture shown in Figure 2.

Table 2 gives the rules for mapping the elements of Figure 1 to TADL.

Table 2. Transformation Rules: Concept Models to Components in TADL

Domain Model Element	TADL Element	Annotation
Entity	Component	*part-of* relation mapped to composite components; *sub-class-of* relation is mapped to a separate component
Data	Attributes	associated with components
Functional Requirement	Service	from *has-property* relation component that provides the service and interface are created; from *requests-property* relation, the component that requests the service and interface are created; a connector is created for every *request-property*
Non-functional	contract	logical expressions - manually done
Constraints	service constraints	logical expressions - manually done

5 Contexts in Components

Contexts are associated with concepts and concepts can be transformed to components. The dependability criteria that involves critical concepts must be satisfied by the components that correspond to those concepts. It is natural that we want the property "if concept X is in context c and concept X is transformed to a component S and M_S is its behavior model[1] then $M_S \models \phi$, where ϕ is the dependability criterion formulated at the domain level". The component will function correctly in a certain context, say c'. Hence we can assert that $\mathbf{ist}(c', M_S)$. We know that $\mathbf{ist}(c, \phi)$. A proof of dependability can be crafted as follows when context c' is more general than context c.
Case 1: Generality of first kind The development method ensures that $c \succeq c'$.

- [1.] Prove $M_S \Rightarrow \phi$ in context c'. Now we can assert $\mathbf{ist}(c', M_S \Rightarrow \phi)$.
- [2.] $\mathbf{ist}(c', M_S)$ is already asserted.
- [3.] By Modes Ponen we infer $\mathbf{ist}(c', \phi)$
- [4.] The property of $c \succeq c'$ implies $\mathbf{ist}(c, \mathbf{ist}(c', \phi))$.

The conclusion is that it is possible to prove in context c that ϕ is valid in context c'.
Case 2: Generality of second kind The development method ensures that $c \gtrsim c'$.

- [1.] $c' = c \sqcup \bar{c}$, $dim(c) \cap dim(\bar{c}) = \emptyset$.
- [2.] The property $c \gtrsim c'$ implies that $\mathbf{ist}(c, \phi) \Rightarrow \mathbf{ist}(c', \phi)$
- [3.] $\mathbf{ist}(c', M_S)$ is already asserted. Still we need to prove $M_S \Rightarrow \phi$ in context c'.

Such a proof is by no means easy. In order to make dependability claim during design we need to have an architecture in which contexts are also first class architectural elements, and carry on the logic of context within the component formalism. Motivated by this we have extended Figure 2 with context inclusion. The extended architecture is quite expressive, in that several families of systems can be developed. Among them *context-aware systems* (CAS) are most important. Ubiquitous computing applications belong to this family. In this section we highlight those mechanisms that are special to CAS architecture.

[1] M_s denotes both the model and behavior.

5.1 Concepts of CAS

Context-aware systems include a heterogeneous environment including people, sensory devices, and actuators. The three major types of interactions are people-system, actuator-system, sensor-system. They are not necessarily independent. Diverse relations of intrinsically complex properties of the environmental entities will produce complex system behavior. Developing such a system, a dependability criteria of it, and assessing it in the system are challenging tasks. Understanding the concepts and following the methodology discussed in earlier sections will certainly maximize the chances of a correct assessment of the system.

We must introduce concepts governing human behavior from cognitive science discipline, concepts on sensors from the electronics and communications engineering, concepts on actuators from mechanical engineering discipline, and concepts on awareness from human psychology. In addition, the development of a context-aware system for a specific application will require concepts pertinent to that application domain. Thus developing a context-aware is more difficult than developing a safety-critical application.

There is a distinction between awareness as interpreted in the study of biological psychology, and awareness that we demand in context-aware systems. In the former case awareness does not necessarily imply understanding. However, in designing computer systems that are expected to be aware, we insist that the system not only understands but remembers (1) its internals such as computing resources, computational states, and policies for state change, and (2) its externals, that may include physical devices and humans. We call the internal monitoring of the system as *self-awareness* and the external monitoring as *context-awareness*. Self-awareness is at the system level and is composed from the policies that guarantee dependable response for every stimulus from users. That is, users must be aware and be convinced that they get dependable service. Hence it is essential to include the concepts on privacy, satisfaction, and expectation with people at the domain level. Such concepts may become part of the dependability criteria.

Moving down to the level of components we need components that correspond to sensors, users, contexts, and actuators. Figure 3 presents a software architecture for context-aware systems. In the architecture we combine users with sensors into *sensor mechanism* component, and combine users with actuators into *environment* component. The *reactivity mechanism* is responsible for performing the reactions and adaptations in the environment. As an example, if the domain is a set of household appliances the sensor mechanism collects data from them and reactivity mechanism controls their behavior subject to dependability criteria. Notice that not all the components shown in Figure 3 are obtainable from domain models. This shows the limitation of domain analysis and the importance of inventing architecture level elements that promote software engineering principles, such as coupling, cohesion, and hiding. Service mechanism implements services that are fed as reactions, and the adaptation mechanism uses security and safety policies from the dependability criteria in different internal and external contexts to trigger appropriate services and reactions. The most important function of CAS is adaptation which we discuss next.

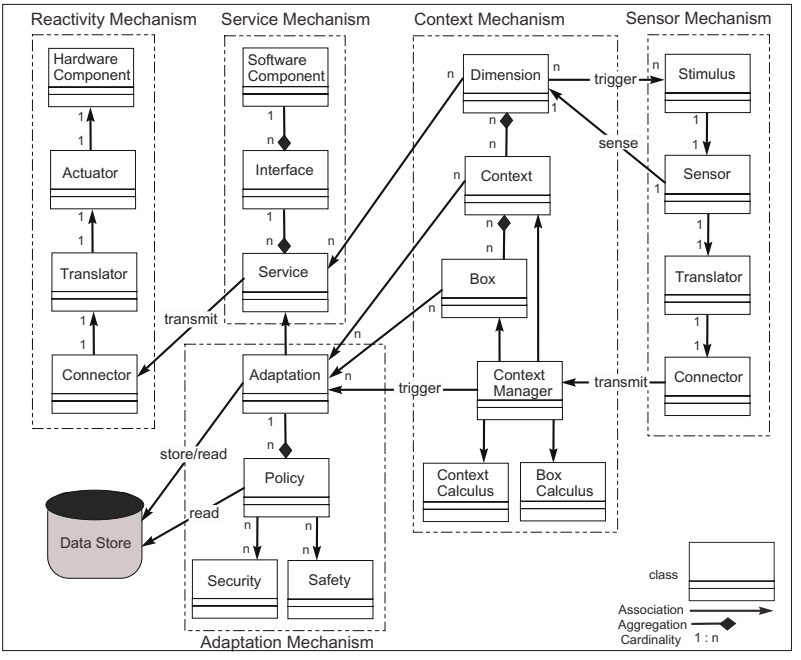


Fig. 3. A software architecture for Context-Aware Systems

5.2 Adaptation Mechanism

The execution pattern of context-aware systems can be modeled using *reactivity*, a relation between awareness and adaptation. A reaction is an event that causes a change in the environment. Context changes stimulate the system to perform new adaptations. Sensors collect context information and send it (after being transformed and built) to context adaptation. The adaptation mechanism is responsible for calculating the suitable reactions for a context. When a context is built, its relevant adaptation reaction is triggered instantaneously. A context may have several possible reactions. In order to achieve predictability, mutually exclusive policies are defined to select the appropriate reaction. We are developing a logic of adaptation, related with the logic of contexts, to formalize service adaptations at different contexts.

Safety, security, and privacy are three special types policies that will regulate adaptations. Safety policies restrict adaptations in order to ensure that the environment receives only reactions that do not damage the environmental entities. Security policies regulate service adaptations and restrict access to context information. Such restriction is essential because context-aware systems are widely used in open environments. Hence, a security policy is defined in terms of *context security* and *adaptation security*. Context security means that a user u will receive information in a context c only if $\mathbf{ist}(c, A_u)$, where A_u is the authorization policy. Adaptation security means that the system will adapt only to the extent for which the user is authorized. As an example, if there is no privacy law governing the individual receiving the service both the user identity and the service type may be made public.

6 Conclusion

In this paper we have made an attempt to piece together many issues and bring out their significance in the development of dependable systems. Research in dependable systems has just begun. As stated in the report [10] a look at the warranty and disclaimer information at the following Web pages Adobe[2], Apple[3], Microsoft[4], and Google[5] should convince us that we have a long way to go before making any of the commonly used software as dependable as we wish them to be.

We are living in an era of ubiquitous computing. Computers affect our lives because we depend on their services. This is a sufficient reason to regard dependable system development as the most important research issue in computing. Dependability criteria lacks a universal definition. It is dependent on the domain in which the software system is to serve. Studying the domain, and analyzing its concepts the dependability criteria should be constructed. Having a dependability criterion in itself is not useful, it must be proved in the system design, implementation, and deployment. This in turn requires a rigorous method that takes us from the domain model to system model and system implementation. In spite of all the care and rigor in building a system that can be verified to satisfy dependability criteria, it is likely that security vulnerabilities in the deployment environment can greatly undermine dependability case. As such a system, taken as a whole in its operational environment, should be protected, audited periodically, and managed well.

References

1. Akman, V., Surav, M.: Steps toward formalizing context. AI Magazine 17(3), 55–72 (1996)
2. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004)
3. Carnap, R.: Meaning and Necessity. Chicago University Press (1947)
4. Christensen, S.R.: Software reuse initiatives at lockheed. CrossTalk 8(5), 26–31 (1995)
5. Clark, H.H., Carlson, T.B.: Context for comprehension. In: Long, J., Baddeley, A. (eds.) Attention and Performance IX, pp. 313–330. Lawrence Erlbaum Associates, Hillsdale (1981)
6. Dowty, D., Wall, R., Peters, S.: Introduction to Montague Semantics. Studies in Linguistics and Philosophy, vol. 11. Springer, Heidelberg (1980)
7. Franklin, M.J.: Challenges in ubiquitous data management. In: Informatics - 10 Years Back. 10 Years Ahead, pp. 24–33. Springer, Heidelberg (2001)
8. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Heidelberg (1999)
9. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: Owl 2: The next step for owl. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), 309–322 (2008)
10. Jackson, D., Thomas, M., Millett, L.: Software for dependable systems: Sufficient evidence? Technical report, Committee on Certifiably Dependable Software Systems, National Research Council (2007)

[2] http://www.adobe.com/products/eula/warranty/
[3] http://www.apple.com/legal/sla/macosx.html
[4] http://www.microsoft.com/windowsxp/home/eula.mspx
[5] http://desktop.google.com/eula.html

11. McCarthy, J., Buvac, S.: Formalizing context (expanded notes). Technical report (1994)
12. Mohammad, M., Alagar, V.: TADL - an architectural description language for trustworthy component-based systems. In: Morrison, R., Balasubramaniam, D., Falkner, K. (eds.) ECSA 2008. LNCS, vol. 5292, pp. 290–297. Springer, Heidelberg (2008)
13. Mohammad, M.S.: A Formal Component-Based Software Engineering Approach for Developing Trustworthy Systems. Phd thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada (2009)
14. Neighbors, J.M.: The Draco approach to constructing software from reusable components. IEEE Transactions of Software Engineering 10(5), 564–574 (1984)
15. Oxford University (Oxford english dictionary), http://www.oed.com/
16. Shortliffe, E.H.: Computer-Based Medical Consultations: MYCIN. Elsevier, Amsterdam (1976)
17. Stanford University: Protege. Stanford University and University of Manchester (2009), http://protege.stanford.edu/
18. Thomason, R.H.: Formal Philosophy: Selected Papers of Richard Montague. Yale University Press (1974)
19. Wan, K.: Lucx: Lucid enriched with context. Phd thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada (2006)

Statistical Methods for Data Mining and Knowledge Discovery

Jean Vaillancourt

UQO, Gatineau, QC J8X 3X7, Canada

Abstract. This survey paper aims mainly at giving computer scientists a rapid bird's eye view, from a mathematician's perspective, of the main statistical methods used in order to extract knowledge from databases comprising various types of observations. After touching briefly upon the matters of supervision, data regularization and a brief review of the main models, the key issues of model assessment, selection and inference are perused. Finally, specific statistical problems arising from applications around data mining and warehousing are explored. Examples and applications are chosen mainly from the vast collection of image and video retrieval, indexation and classification challenges facing us today.

1 Introduction

Data mining for the purpose of knowledge discovery has been around for decades and has enjoyed great interest and a flurry of research activity in just about every field within the scientific community. The purpose here is to broach the topic from a mathematician's perspective with a view towards understanding what exactly can be said and concluded, when using some of the more sophisticated statistical tools to extract knowledge from large data bases.

The statistical techniques used currently in data mining practice often come from the great statistical toolbox built out of problem solving issues arising in other fields and taught in the standard science curriculum. They require several hypotheses to be checked and tested. These hypotheses reflect in large part the basic character of the data under study and it is vital to assess whether or not these hypotheses are valid in order to draw any inference from our data mining endeavor. This is especially true when using the more sophisticated and complex statistical estimators and tests.

In this survey paper I first give a rapid bird's eye view of the main statistical methods used in order to extract knowledge from databases comprising various types of observations. After touching briefly upon the matters of supervision, data regularization and a brief review of the main models, the key issues of model assessment, selection and inference are perused. Finally, specific statistical problems arising from applications around data mining and warehousing are explored. Ten years of collaboration with experts in image and video retrieval, indexation and classification, as well as in formal concept analysis, color my choice of examples and applications; however, the statistical methods discussed are, now as ever, universal.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 51–60, 2010.

In spite of this ambitious program, the style of this paper is purposefully light and equations will be kept to a minimum. Extensive references to the literature should yield ample compensation for this shortcoming, as the goal here is to supply the mathematical intuition behind the choice of methods rather than the precise formulas which can be found elsewhere, for instance in the few main references brought forward next. No claim is made as to the novelty of the statistical methods described here. The originality of the paper lies instead in the choice and presentation of the material, inasmuch as it displays the aforementioned heavy personal bend towards applications in imaging.

The main sources used in producing this survey comprise first and foremost, the excellent book [1] of Hastie, Tibshirani and Friedman. Friedman's take on the link between data mining and statistics [2] provides a complementary treatment of the interface between the two fields as it stood fourteen years ago, a treatment that is still relevant today, particularly in the light of Hand's paper [3] and Friedman's comments [4] on classifier methodology. The emphasis in these sources is put on supervised learning as it affords a wealth of existing statistical techniques to be used in specific applications and to be compared in performance under precise conditions.

To understand the distinction between supervised and unsupervised learning, a simple classification example from the field of content-based image search may be helpful. The goal pursued in this field is the automatic identification and labeling of images containing a variety of objects, some of interest and some not, as well as noise from several sources linked to distance, media related distorsion or even physical features of the image collection device. The ultimate goal is of course the retrieval of images containing objects in a prespecified class with some degree of success. In [5] Sarifuddin et al. propose a framework whereby certain objects of interest are sought within a structured database of pictures, using similarity measures based solely on the color characteristics within the image. Here the target population is the database itself, the search is done on the whole database but interest lies on the top five to ten hits, so fluctuations are observed between the competing search algorithms. A training subset of images is used for computational purposes and learning is achieved through similarity measurements with the rest of the database. The algorithms proposed there have the feature of predicting a new top hit (the outcome of the learning experiment) whenever the training set (the input) is changed, with some measure of the error, thus yielding opportunities for making credible and scientifically defensible inferences on the whole database. This last ability for meaningful prediction expresses the presence of supervision and allows for a probability (or sample) space to be defined with rigor. Many image, video and metadata bases possess these features : collections of highway camera readings of licence plates on rear views of cars and lists of persons of interest in police work, standardized DNA databanks, virtual museum collections, topographic maps and handwritten zipcodes on surface mail are but a few examples. They tend to be well structured for searching, grow (relatively) slowly and require frequent access.

By contrast, just attempt searching the web (or some other database not specifically structured with content-based image search in mind) through your favorite search engine by way of some bit of text (in image search parlance this is called semantic information). An excellent survey of the most popular systems including Diogenes, ImageRover, WebSeek and Atlas WISE can be found in Kherfi, Ziou and Bernardi [6]. The result is a collection of images that do not constitute an outcome in the inferential sense above, since the database is for all intent and purposes infinite and the sample space is defined with a new ambiant distribution upon every iteration of the algorithm as every new search receives no quantifiable predictive benefit from the previous ones. This inability to predict new outcomes with statistical measurements of accuracy typifies unsupervised learning and data mining in general.

Unsupervised learning offers limited avenues to measure rigorously the validity of inferences or to compare learning methods with some degree of scientific credibility. By enriching association rule analysis with inherent graphical and hierarchical structures, formal concept analysis (FCA) (Ganter and Wille [7]) provides a stepping stone towards this end through the systematic process of generating ontologies. Our main reference for FCA here is Valtchev, Missaoui and Godin [8]. It is as much a choice of convenience (it is still current as an overview) as a personal one.

The next section reviews the various choices afforded to the data analyst when selecting a statistical learning model. The matters of identifying the possibility of supervision, the presence in the data of linearity, the need for grouping or regularization, and finally the key issues of model assessment and selection, as well as that of inference and testing, are briefly touched upon. The last section delves into ancillary issues relating to the organization of data into shapes amenable to statistical treatment, with an eye on image and video sources for illustrating purposes.

2 Statistical Teaser

Let us begin by reviewing some basic definitions and notation used throughout this paper. The terminology is described in the context of supervised learning, but we will keep the same terms when switching to unsupervised learning, even though this context will require some clarification when interpreting the results.

Good statistics ideally starts with good data collected with a view to an end. Data usually comes in the form of a set of input (independent or predictor) variables, which we control or are able to measure; and a set of output (dependent or response) ones, which are observed. Some of them will be quantitative measurements, some not (called factors or qualitative or even categorical variables). Throughout this paper, the variables will be noted by uppercase letters (X for input and Y for output, with real, vector, matrix or even functional values as required by the context) and their values by lowercase ones (similarly).

Predicting quantitative outputs is called regression and qualitative outputs, (statistical) classification. In both cases, whatever the technique of choice (more

on this later), supervised learning is achieved through a predictive model of the form $Y = f(X, \epsilon)$ where ϵ is a random error and f is an unknown function to be chosen or estimated from the observed (raw) data (x_i, y_i) for $i = 1, 2, \ldots, n$. The function will usually be selected according to some optimization (scoring) principle, from a simple family allowing the algebraic or numerical isolation of the error.

Keep in mind that while a classification problem can always be reformulated as a regression one using indicator (dummy) variables, their number (one per class of values per variable) grows very rapidly. As a result many techniques are specially devised to deal with estimation in that context. We shall nevertheless focus on the regression side of statistics in this section in order to keep it short.

The classical least squares linear regression known to all is just the case $f(X, \epsilon) = g(X) + \epsilon$ with g in a restricted set of nice functions and estimated to some $\hat{g}$ that minimizes the sum of squares or errors $\sum_{i=1}^{n} \|y_i - g(x_i)\|^2$ as a function of g. Here $\|\cdot\|$ denotes some appropriate norm. The basic examples where g is itself a linear function or a polynomial (possibly in many variables, in which case the x_i's are vector, matrix or even function valued) are the most commonly used models around, even when the input is time dependent (where the theory is already rich and complex from a mathematical point of view, see Solo [9]) or time and space dependent, as are some current models for video segmentation (see for instance Cremers [10]). The target value in this case is simply the conditional expectation $E(Y|X)$.

This basic idea has been expanded to richer families of functions g than linear ones (such as piecewise polynomials, splines, wavelets or the directional mappings used in projection pursuit regression) but we shall not dwell on these methods in this least squares context since their implementation challenges are not key to our purpose here. All of these so-called regularization methods aim at approximating the true (unknown and nonlinear) function g of the input data in order to ensure a better fit of the model to the (usually highly nonlinear) output data. The interested reader may read Wahba [11] for more on splines, Daubechies [12] on wavelets and our main reference [1] for a thorough review of projection pursuit and the statistical side of neural networks in general.

Another direction for development which has shown great results in several areas of application including satellite imaging is the use of mappings $f(X, \epsilon)$ that are not linear in ϵ. Recall that segmentation algorithms are a crucial part of the automatic systems used in modeling and processing image data. These algorithms parse out each image into smaller ones (the content of which tends to be simpler), thus enhancing the discriminating power of the searching tools by increasing the spread (or variety) of visual features extracted and used for identification. These features normally include color, texture and shape, as well as some dynamical measurements in the case of video databases. The statistical comparison of unsupervised segmentation algorithms on structured image databases was initiated by Graffigne et al. in [13] and [14]. The two main classes of methods studied were the bayesian ones (in the variational context of energy minimization later used to great effect by Bentabet et al. in [15] and by Jodouin

et al. in [16]), pionneered in the works of Grenander [17] and Geman and Geman [18] ; and hierarchical Markov random field based methods (first used for image analysis by Besag in a series of papers starting with [19] through to [20]). In both cases the methods were used in order to decrease the large size of the optimization problem posed by image segmentation — to see just how slow the process is, even when a Gibbs sampler is used to accelerate the process, read Gibbs [21]. In much of the research done on image segmentation and restoration, the noise source is assumed to affect the image additively. This is simply not the case in SAR imaging, since the noise (speckle) does not behave additively — this is borne out both by analyzing the shortcomings of linear methods like Fourier transforms (see DeGraaf [22]) and by noticing the (undesirable) heteroscedastic behavior of the residuals when modeling with additive noise, a tell tale sign of lack of fit from a statistical point of view. Proper modeling calls for nonlinear dependency in the noise, like the multiplicative noise used for instance in [15] and [16] where very good fit is attained. This last approach, when combined with hierarchical (multi-resolution) Markov methods, is state of the art and also affords rigorous mathematical tracking.

Refinenements of this basic least squares regression method abound in the literature and lead to the construction of alternative (often better fitted) estimates to the above $\hat{g}$. The target value in most of the classes of alternatives mentioned below will usually no longer have the simple closed form of a conditional expectation and require some numerical effort in order to reach an approximate value.

As a first refinement, the square function in the previous example can be replaced by some other loss function, for instance, through the addition of a penalty term like the weight decay used in projection pursuit, neural networks or shrinkage methods (like ridge regression), if the outputs are quantitative.

Explicit probabilistic modeling offers another collection of techniques. If the data collection is to be repeated often and rapidly, the observer may choose to weight the data unequally according to some a priori distribution on the outputs (as in the bayesian approach, useful with both types of outputs) or according to a well chosen mixture of distributions on the whole data (which has shown great success in many applications including image selection, as in Bouguila, Ziou and Vaillancourt [23] where the bayesian framework and a clever choice of mixtures are combined to great effect). Detecting the presence of a mixture in data has been adressed in Walther [24]. Mixture modeling offers a natural framework to locate common features like data clusters and to discriminate between classes of output values.

Likelihood based methods constitute a third refinement (actually more of an alternative) to classical least squares regression by imposing a statistical framework into the model $Y = f(X, \epsilon)$ from the get-go. They constitute a large body of statistical literature (see Severini [25] for a good overview), give meaningful (statistically interpretable) results even with small sample sizes, are based on the likehood principle respected even by bayesian posterior decision analysis (see section 4.4 in Berger [26]), and their asymptotics are tractable rigorously (see Prakasa Rao [27]). They perform very well in complex mathematical contexts

like that of computer vision, as displayed in Amit and Geman [28] and further in Amit and Trouvé [29].

A fourth refinement consists in preselecting target areas in the space of input values and applying some local method, such as local regression, density estimation and the many methods relying on measures of similarity, nearest neighbors, clustering or kernel functions at each point of interest in that space. These methods require particular care and attention when used on high dimensional data as they then tend to show much sensitivity to additional data and lose both accuracy (minimal bias) and precision (minimal spread) against the other classes of methods. Combining them with judiciously chosen distributional restrictions from those in the previous paragraphs is the usual way out. This approach has been used successfully in the context of image retrieval, indexation and sorting. For example, the importance of statistical mixtures of Dirichlet distributions in computerized image searches was first brought forward in [23] after a series of papers (listed therein) on particular aspects of the subject. Similarity measures have also played a central role in proper mathematical frameworks for image selection and the reader should consult Missaoui et al. [30] and Sariffudin et al. [5] for some simple and convincing examples. Finally on this issue, everything you ever wanted to know about the basic statistical aspects of kernel estimators can be found in Devroye [31]. On the computational side of things, these methods often (but not always, see [23] and [30]) turn out to be impracticable because of excessive cost.

The fifth and final class of alternatives (and the most commonly handled in computer science practice) require some systematic selection of a subset of the input data (for instance, when using shrinkage); of a boundary within the data to break the problem down to smaller size (using separating hyperplanes to find good linear boundaries or support vector machines to find nonlinear ones or else tree-based methods to break the space into a few partially ordered blocks); or of a small subset of certain combinations of the whole input data (principal component regression, projection pursuit and neural network methods fall in this class). All three approaches work by decreasing the size or the dimensionality of the space of input variables while preserving what the observer believes to be the core features of interest. With extremely large databases, they tend to be the wise way to go.

Armed with this rich collection of models, the experimenter now comes to the matter of model selection and assessment. Selection consists in estimating the performance of each model contemplated with the purpose of choosing the best; assessment, in estimating the predictive power of each model on new data. The key issue at this point is not to decide right away whether or not your model of choice is the best for the data at hand (no model ever outperforms all others in all situations anyway) but rather to check how good is your data to start with. A good training set often has more impact on the quality of the results of the inferential selection and assessment process than the sophistication and complexity of the models chosen for comparative purposes.

If your data set of interest is very large, common practice consists in separating it in three parts, one serving the purpose of the training set, one used to select the model of choice through a validating estimation of the prediction error (the validation set) and one for the final assessment of the predictive ability of the chosen model on new data (the test set). This allows for accurate measurements of bias and variability. When this is not the case, one needs to generate pseudo-observations to compensate. This is usually done by way of resampling techniques known as jackknifing and bootstrapping (see Efron [32] for the simplest contexts), as well as the derivatives of the bootstrap known as bagging (bootstrap aggregating, due to Breiman [33]) and boosting (see [1]). These last two methods are amenable to nonlinear function estimation and any choice of loss function. They display remarkable performance in tests and benchmarks against most other methods mentioned thus far. They also tend to be robust against many distributional alternatives, a reassuring characteristic when dealing with large heterogeneous data sets. The basic idea behind their use is that averaging amongst several predictors should decrease the variability of the results while maintaining bias to a minimum. Applying the various models to several data sets will of course make the results more credible, especially when combined through the construction of a random forest, (a double randomization technique involving both bootstrapping of the data and random selection of the subset of most interesting variables) which can be rigorously analyzed (see Breiman [34]).

Model selection now becomes a matter of chosing a loss function relevant to the nature of the data (usually a measure of distance like the sum of squares or a measure of entropy like the sum of log-likelihoods) and then estimating the parameters associated with each model under purview in order to minimize (at least approximately) the corresponding expected loss. A trade off between accuracy (small bias and spread) and parcimony (as few parameters as possible) will usually be included in the loss function or the optimization scheme itself. The importance of parcimony is convincingly made by Besse et al. [35] and we concur with them that simpler models, whether they involve neural networks, mixtures, support vector machines or any other sophisticated tools, combined with resampling, generally constitute a better choice from both efficiency and reliability standpoints, than complex interpretative ones with large numbers of parameters to be estimated.

3 Unsupervised Learning and Statistics: Some Challenges

We now turn to the context of unsupervised learning and data mining, urging the reader to consult Besse et al. [35] as well as chapter 14 of Hastie, Tibshirani and Friedman [1]. At the risk of being repetitive, keep in mind that any information held by the scientist about the data prior to mining (or snooping), must be incorporated in the experimental design leading to data collection in order to have some hope of checking statistical hypotheses. These informations are usually available in the presence of supervision and at least some of the statistical hypotheses can be checked.

In (unsupervised) data mining one usually cannot afford this level of control over data collection, since the data warehouses are usually assembled before the experimentation is devised, a pity. The choice of the training set is often the only latitude left at our disposal and it should be done with some care and attention towards ensuring that some statistical inference can be made. Nevertheless, as long as the conclusions drawn from the experiment are formulated with the proper reserve, selecting methods with a proven record of quality remains the sensible thing to do.

Many of the early developments in data mining methodology stemmed from incursions into exploratory data analysis (EDA) that predated data mining and were based on key ideas already in the statistical literature at the time. These incursions were championned in the mid sixties simultaneously, independently and along completely different methodological approaches by Tukey (see [36] for the history of this branch of EDA) using (and reinventing) robust statistics and by the French school of EDA inspired by classical geometry and initiated with the proposal by Escofier (in his 1965 doctoral thesis) of correspondance analysis as we know it today (see Benzécri [37] for a history of this branch of EDA).

For a statistician, unsupervised learning consists in estimating the probability distribution of the input variable X (usually valued in a space of large dimension) based on the (vector, matrix or even functional valued) observations $x_1, x_2, \ldots, x_n$. There is no output variable since the focus here is understanding the data and discovering pertinent subsets rather than predicting outcomes.

In low dimensions, non parametric density estimators (see Devroye [31]) will usually provide sufficient insight into the data to satisfy the user. The large dimension of the space in data mining makes these density estimators unreliable; however, they can still be used to get estimates for the one dimensional marginal distributions (margins) of the input X. Once the margins have been estimated, the problem at hand becomes equivalent to the determination of an appropriate copula for the distribution of X given its margins. A copula is a multidimensional distribution on the unit cube of the space of values of X with uniform one dimensional margins. This provides a first strategy to extract knowledge from the observations, since copulas are currently well researched (if at times controversial, see Genest and Rémillard [38]). As copulas comprise all the information within a distribution given the margins, they are a powerful tool indeed but determining them explicitly in high dimension remains difficult for now.

One gets around this issue by finding instead the most frequent values (statistical modes) of X within the data base, since a large enough number of them will cover the most significant regions of the distribution. This approach greatly reduces dimensionality, is easier to implement and has become the very popular technique of association rule mining. Alternatively, one bypasses the probability model and, using a measure of distance or similarity on the data, looks for agregated clouds of data points through one of the many clustering algorithms available. Again one is faced with classical methods that quickly find their limits in high dimensional space unless supported by one of the data reduction techniques mentioned in section 2 (our fifth class of alternatives). For

example, Bouguila [39] combines clustering with mixtures to generate a rich class of models and then uses marginal likelihood for succesful model selection.

The first attempts at setting the probability distribution on the (oriented) relations between input observations instead of the observations themselves led to the creation of statistical implicative analysis, recently surveyed by Gras and Kuntz in [40]. Since the number of such relations grows like the square of the size of the database, it suffers the same challenges as unaided clustering algorithms do. The twin needs to enrich the set of relations and to reduce the speed of growth of the pertinent or good subsets leads to formal concept analysis (FCA). According to Valtchev, Missaoui and Godin [8], FCA has demonstrated cost-effectiveness, adaptability and user-friendliness in a variety of settings. Incorporating statistical structure to FCA will likely be the next stage to making it step from a very useful mathematical tool for structuring knowledge, to a privileged methodology for drawing rigorous inference from large and complex data warehouses.

References

1. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning. Springer Series in Statistics (2001)
2. Friedman, J.H.: Data Mining and Statistics: What's the Connection? Keynote presentation at 29th Symposium on Interface: Computer Science and Statistics (1997), http://www-stat.stanford.edu/~jhf/
3. Hand, D.: Classifier technology and the illusion of progress. Statist. Sci. 21(1), 1–14 (2006)
4. Friedman, J.H.: Comment on classifier technology and the illusion of progress. Statist. Sci. 21(1), 15–18 (2006)
5. Sarifuddin, M., Missaoui, R., Vaillancourt, J., Hamouda, Y., Zaremba, M.: Analyse statistique de similarité dans une collection d'images. Revue des Nouvelles Technologies de l'Information 1(1), 239–250 (2003)
6. Kherfi, M.L., Ziou, D., Bernardi, A.: Image retrieval from the world wide web: issues, techniques and systems. ACM Computing Surveys 36(1), 35–67 (2004)
7. Ganter, B., Wille, R.: Formal concept analysis, mathematical foundations. Springer, Heidelberg (1999)
8. Valtchev, P., Missaoui, R., Godin, R.: Formal concept analysis for knowledge and data discovery: new challenges. In: Proc. Second Int. Conf. Formal Concept Analysis, Sydney, Australia, pp. 352–371 (2004)
9. Solo, V.: Topics in advanced time series analysis. Lecture notes in mathematics, vol. 1215, pp. 165–328. Springer, Heidelberg (1986)
10. Cremers, D.: Bayesian approach to motion-based image and video segmentation. In: Jähne, B., Mester, R., Barth, E., Scharr, H. (eds.) IWCM 2004. LNCS, vol. 3417, pp. 104–123. Springer, Heidelberg (2007)
11. Wahba, G.: Spline models for observational data. SIAM, Philadelphia (1990)
12. Daubechies, I.: Ten lectures on wavelets. SIAM, Philadelphia (1992)
13. Graffigne, C., Heitz, F., Perez, P., Preteux, F.J.: Hierarchical Markov random field models applied to image analysis: a review. In: Proc. SPIE, vol. 2568, pp. 2–17 (1995)
14. Graffigne, C.: Stochastic modeling in image segmentation. In: Proc. SPIE, vol. 3457, pp. 251–262 (1998)

15. Bentabet, L., Jodouin, S., Ziou, D., Vaillancourt, J.: Road vectors update using SAR imagery: a snake-based approach. IEEE Trans. on Geoscience and Remote Sensing 41(8), 1785–1803 (2003)
16. Jodouin, S., Bentabet, L., Ziou, D., Vaillancourt, J., Armenakis, C.: Spatial database updating using active contours for multi-spectral images: application with Landsat 7. ISPRS J. of Photogrammetry and Remote Sensing 57, 346–355 (2003)
17. Grenander, U.: Lectures in pattern theory, vol. I, II and III. Springer, New York (1981)
18. Geman, D., Geman, S.: Stochastic relaxation, Gibbs distributions and the bayesian restoration of images. IEEE Trans. Pattern Anal. Math. Intell. 6(6), 721–741 (1984)
19. Besag, J.: Spatial interaction and the statistical analysis of lattice systems. J. Roy. Statist. Soc., B 36, 192–236 (1974)
20. Besag, J.: On the statistical analysis of dirty pictures. J. Roy. Statist. Soc., B 48, 259–302 (1986)
21. Gibbs, A.L.: Bounding the convergence time of the Gibbs sampler in Bayesian image restoration. Biometrika 87(4), 749–766 (2000)
22. DeGraaf, S.R.: SAR imaging via modern 2-D spectral estimation methods. IEEE Trans. on Image Processing 7(5), 729–761 (1998)
23. Bouguila, N., Ziou, D., Vaillancourt, J.: Unsupervised learning of a finite mixture model based on the Dirichlet distributions and its applications. IEEE Trans. Image Processing 13(11), 1533–1543 (2004)
24. Walther, G.: Multiscale maximum likelihood analysis of a semiparametric model, with application. Ann. Stastist. 29(5), 1297–1319 (2001)
25. Severini, T.: Likelihood methods in statistics. Oxford Univ. Press, Oxford (2001)
26. Berger, J.O.: Statistical decision theory and bayesian analysis. Springer, Heidelberg (1980)
27. Prakasa Rao, B.L.S.: Asymptotic theory of statistical inference. John Wiley, Chichester (1987)
28. Amit, Y., Geman, D.: A computational model for visual selection. Neural Computation 11, 1691–1715 (1998)
29. Amit, Y., Trouvé, A.: POP: Patchwork of parts models for object recognition. Intern. J. Comp. Vision 75(2), 267–282 (2007)
30. Missaoui, R., Sarifuddin, M., Vaillancourt, J.: Similarity measures for an efficient content-based image retrieval. In: IEE Proc. Vision, Image and Signal Processing, vol. 152(6), pp. 875–887 (2005)
31. Devroye, L.: A course in density estimation. Birkhauser Verlag, Basel (1987)
32. Efron, B.: The jackknife, the bootstrap and other resampling plans. SIAM, Philadelphia (1982)
33. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
34. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
35. Besse, P., Le Gall, C., Raimbault, N., Sarpy, S.: Data mining et statistique, avec discussion. Journal de la Société Francaise de Statistique 142, 5–35 (2001)
36. Tukey, J.W.: Exploratory data analysis. Addison-Wesley, Reading (1977)
37. Benzécri, J.P.: Histoire et préhistoire de l'analyse des données. Dunod (1982)
38. Genest, C., Rémillard, B.: Comments on T. Mikosh's paper Copulas: tales and fact. Extremes 9, 27–36 (2006)
39. Bouguila, N.: A model based approach for discrete data clustering and feature weighting using MAP and stochastic complexity. IEEE Trans. Knowledge and Data Engineering 21(12), 1649–1664 (2009)
40. Gras, R., Kuntz, P.: An overview of the statistical implicative analysis (SIA) development. Studies in computational intelligence, vol. 127, pp. 11–40 (2008)

Formal Concept Analysis of Two-Dimensional Convex Continuum Structures

Rudolf Wille

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

Abstract. This paper offers an approach of developing an order-theoretic structure theory of *two-dimensional convex continuum structures*. The chosen approach is based on convex planar continua and their subcontinua as primitive notions. In a first step convex planar continua are mathematized and represented by ordered sets. In a second step '*points*' are deduced as limits of continua by methods of *Formal Concept Analysis*. The convex continuum structures extended by those points give rise to *complete atomistic lattices* the atoms of which are just the smallest points. Further research is planned to extend the approach of this paper to *higher dimensional conti-nuum structures*.

Contents

1 Introduction

Philosophically, a *continuum* is in its total a phenomenon which preserves its whole across possible cuts and borders. In particular, space and time can be understood as such continua. According to Aristotle in his physics lecture, a *continuum* is an unlimited complete continuity which cannot be dismantled into indivisible parts ([Ar95]; p.616). This paper offers an approach of developing an order-theoretic structure theory of *two-dimensional convex continuum structures* which can be obtained by mathematizing *convex planar continua* in the sense of Aristotle. The one-dimensional case has been published in [Wi08].

2 Convex Planar Continua

Phenomenologically, a *convex planar continuum* is an unlimited convex continuous plane which can always be divided into two convex planar subcontinua by a so-called (*unlimited*) *straight linear cut*. Two straight linear cuts divide a convex planar continuum into three or four convex planar subcontinua; those pairs of cuts are called *"parallel cuts"* in the first case and *"crossing cuts"* in the second case.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 61–71, 2010.

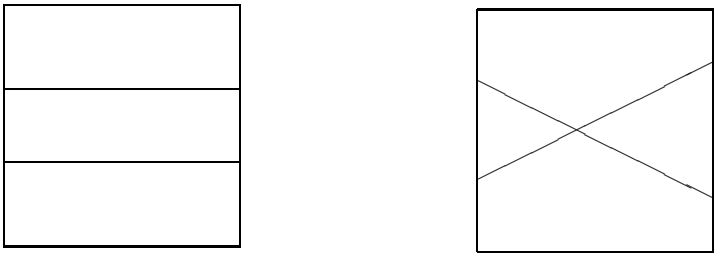


Fig. 1. The diagrams show two convex planar continua each represented by the inner area of a rectangle: the first is divided into three subcontinua by two parallel cuts and the second into two opposite pairs of subcontinua by two crossing cuts

In the case of two parallel cuts, there exist always three further straight linear cuts parallel to the given two cuts so that each of the three subcontinua is divided by one of the three further cuts into two smaller subcontinua. This has as consequence that between two parallel cuts there are always infinitely many further parallel cuts and therefore also infinitely many *"parallel"* subcontinua.

In the case of two crossing cuts, there exist always two further straight linear cuts so that each of those cuts divides one of the two opposite pairs of the four subcontinua into four further subcontinua, but does not cut into the other opposite pair of subcontinua. This procedure of dividing two opposite pairs of four subcontinua can also be infinitely repeated such that each opposite pair of subcontinua is divided into infinitely many further subcontinua.

An infinite collection of straight linear cuts in which every two cuts are parallel can be understood as an isomorphic copy of a linear continuum as described in [Wi08]. An infinite collection of crossing cuts of which every three cuts divide the underlying continuum into three pairs of opposite subcontinua has a *center* through which all cuts of the collection are passing. Such a center shall be called an *"infinitesimal hole"*.

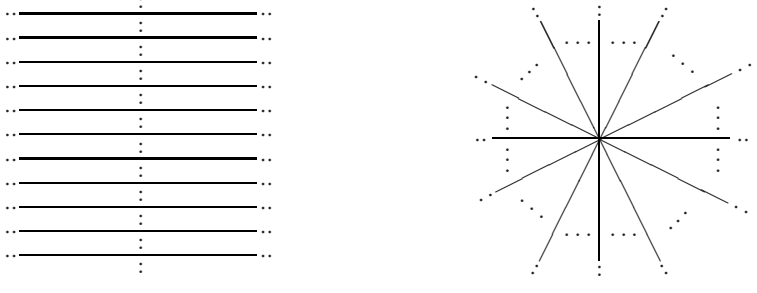


Fig. 2. The left diagram represents a subcontinuum structured by a dense bundle of parallel cuts in a convex planar continuum and the right diagram represents a subcontinuum structured by a dense bunch of cuts all of which pass through the same infinitesimal hole of the underlying convex planar continuum

A *convex planar continuum* shall fulfill, in addition to the already explained properties, the following three axioms:

1. Each convex subcontinuum contains infinitely many infinitesimal holes.
2. For every two infinitesimal holes h_1 and h_2, there exists exactly one straight linear cut which is passing through h_1 and h_2.
3. For each straight linear cut and each infinitesimal hole h through which the cut c is not passing, there exists exactly one straight linear cut d which passes through h and is parallel to c.

Each convex planar subcontinuum $\tilde{C}Sub2$ is uniquely characterized by the entirety $Cut(\tilde{C}Sub2)$ of all straight linear cuts which do not divide the subcontinuum. Obviously, such a subcontinuum $\tilde{C}Sub2$ is contained in another convex planar subcontinuum $\tilde{C}\hat{S}ub2$ if and only if $Cut(\tilde{C}\hat{S}ub2)$ is contained in $Cut(\tilde{C}Sub2)$.

3 Two-Dimensional Convex Continuum Structures

A *"convex planar continuum"* $\tilde{C}S2$ can be properly mathematized to become a suitable ordered set $\underline{CS2} := (CS2, \leq)$ as follows: $CS2$ comprises all convex planar subcontinua $CSub2$ together with itself as subset, where the order relation $\leq$ between those elements is defined by $CSub2 \leq \hat{C}Sub2$ if and only if $Cut(\hat{C}Sub2) \subseteq Cut(CSub2)$ and $CSub2 \subseteq CS2$. Mathematically, such ordered set $\underline{CS2}$ is called a *"Two-Dimensional Convex Continuum Structure"*.

There are two basic types of mathematical quotient structures of $\underline{CS2}$ (as already indicated in Fig. 2):

1. the type $\underline{CS2}||$ of a *"dense bundle of parallel cuts"* and
2. the type $\underline{CS2}\mathsf{X}$ of a *"dense bunch of cuts passing through the same infinitesimal hole"*.

Definition 1. A *"two-dimensional convex continuum structure"* is mathematically defined as an infinite ordered set $\underline{CS2} := (CS2, \leq)$ satisfying the following conditions:

(1) $\underline{CS2}$ is a $\bigvee$-semilattice with greatest element $\mathbf{1}$ and without a smallest element;
(2) the $\wedge$-irreducible elements of $\underline{CS2}$ form infinitely many (unorganized) pairs $\{\acute{C}_j, \grave{C}_j\}$ of disjoint linearly ordered dense chains without greatest and smallest element, where $\acute{c}_j \vee \grave{c}_j = \mathbf{1}$ for all $\acute{c}_j \in \acute{C}_j$ and $\grave{c}_j \in \grave{C}_j$ $(j \in J)$;
(3) each such pair $\{\acute{C}_j, \grave{C}_j\}$ has infinitely many *"fitting pairs"* $\{c_{jk}^{\dashv}, c_{jk}^{\vdash}\}$ with $c_{jk}^{\dashv} \in \acute{C}_j$ and $c_{jk}^{\vdash} \in \grave{C}_j$ (also called *"fitting cuts"*) each of which has $\mathbf{1}$ as its supremum where the intervals $[c_{jk}^{\dashv}, \mathbf{1}]$ and $[c_{jk}^{\vdash}, \mathbf{1}]$ are maximal according to the property that $c_{jk}^{\dashv}$ and $c_{jk}^{\vdash}$ have no common lower bound in $\underline{CS2}$; furthermore, $c_j^{\dashv} \wedge c_j^{\vdash} = d_j^{\dashv} \wedge d_j^{\vdash}$ implies $c_j^{\dashv} = d_j^{\dashv}$, $c_j^{\vdash} = d_j^{\vdash}$ for all $c_j^{\dashv}, d_j^{\dashv} \in \acute{C}_j$ and $c_j^{\vdash}, d_j^{\vdash} \in \grave{C}_j$ $(j \in J)$;
(4) two different fitting pairs $\{c_j^{\dashv}, c_j^{\vdash}\}$ and $\{d_j^{\dashv}, d_j^{\vdash}\}$ give rise to three $(||)$ or four (X) further elements each two of which have no common lower bound; in the three element case $(||)$, there are three further pairs of elements so that each of the three elements is the join of two from the six further elements;

in the four element case (X), there are four further pairs of elements so that each of the four elements is the join of two from the eight further elements;

(5) the basic elements described in (4) are infinitely extended by further fitting pairs leading to infinite refinements to get *"dense bundles of parallel fitting pairs"* and *"dense bunches of fitting pairs through the same infinitesimal hole"*; in the dense bundle case (||), parallel fitting pairs are refined in *one dimension*; in the dense bunch case (X), centring fitting pairs are refined in pathing through a *fixed center* (cf. Fig. 2);

(6) the term *"infinitesimal hole"* stands for the infinitely dense bunches of all fitting pairs pathing through a fixed imaginary location;
for every two infinitesimal holes h_1 and h_2, there exists exactly one fitting pair of elements which is pathing through h_1 and h_2;
for every fitting pair c and each infinitesimal hole through which the fitting pair is not pathing, there exists exactly one fitting pair d which passes through h and is parallel to c.

◊

Example 1. The ordered two–dimensional real vector space $\mathbb{R}^2$ gives rise to the *two-dimensional convex continuum structure* $\underline{C}_{\mathbb{R}^2} := (C_{\mathbb{R}^2}, \subseteq)$. The set $C_{\mathbb{R}^2}$ has the non-empty open convex subsets of $\mathbb{R}^2$ as elements. In this ordered set, the greatest element is the set $\mathbb{R}^2$ and the supremum operation is determined by the open convex hull of the set-theoretic union. The $\wedge$-irreducible elements of $C_{\mathbb{R}^2}$ are the open half-plains having an unlimited straight line as boundary; two such half-plains with the same unlimited straight line as boundary form a "fitting pair" $\{c_j^{\dashv}, c_j^{\vdash}\}$ of elements in $C_{\mathbb{R}^2}$.

4 Concept Lattices Derived from Ordered Sets

Formal Concept Analysis [GW99] shall be activated now to make mathematically explicit that points, according to Aristotle [Ar95], can be understood as limits of continua; such limits cannot be part of continua. The extension by points will be performed by using a general method of Formal Concept Analysis which mathematically establishes the transfer from ideas to concepts in the sense of the *structure-genetic psychology* of Jean Piaget [Pi59]. The mathematization of this transfer is grounded on ordered sets $\underline{C} := (C, \leq)$ of preconceptual 'ideas' (cf. [SW86]). For analysing the general method we have to refer to the Basic Theorem of Concept Lattices which therefore shall be recalled here (see also [Wi08]):

Basic Theorem on Concept Lattices. [Wi82] *Let* $\mathbb{K} := (G, M, I)$ *be a formal context. Then* $\underline{\mathfrak{B}}(\mathbb{K})$ *is a complete lattice, called the* concept lattice *of* $\mathbb{K}$*, whose infima and suprema can be described as follows:*

$$\bigwedge_{t\in T}(A_t, B_t) = (\bigcap_{t\in T} A_t, (\bigcup_{t\in T} B_t)^{II}), \quad \bigvee_{t\in T}(A_t, B_t) = ((\bigcup_{t\in T} A_t)^{II}, \bigcap_{t\in T} B_t).$$

In general a complete lattice L *is isomorphic to* $\underline{\mathfrak{B}}(\mathbb{K})$ *if and only if there exist mappings* $\gamma : G \to L$ *and* $\mu : M \to L$ *such that* γG *is* $\bigvee$-dense *in* L (*i.e.*

$L = \{\bigvee X \mid X \subseteq \gamma G\}$), μM is $\bigwedge$-dense in L (i.e. $L = \{\bigwedge X \mid X \subseteq \mu M\}$), and $gIm \iff \gamma g \leq \mu m$ for $g \in G$ and $m \in M$; in particular, $L \cong \underline{\mathfrak{B}}(L, L, \leq)$ and, if the set $J(L)$ for all $\bigvee$-irreducible elements is bigvee-dense in L and the set $M(L)$ for all $\bigwedge$-irreducible elements is bigwedge-dense in L, then we have that $L \cong \underline{\mathfrak{B}}(\mathbb{K})$.

The process of concept building models formal objects by filters of $\underline{C}$ and formal attributes by ideals of $\underline{C}$ (cf. [SW86]). A *filter* of $\underline{C}$ is a non-empty subset F of C, for which $a \in F$ and $a \leq b$ imply $b \in F$ and $a, c \in F$ guarantees the existence of some $d \in F$ with $d \leq a, c$; an *ideal* of $\underline{C}$ is dually defined to the filter[1]. This modelling leads to the derived context $\mathbb{K}(\underline{C}) := (\mathfrak{F}(\underline{C}), \mathfrak{I}(\underline{C}), \Delta)$ for which $\mathfrak{F}(\underline{C})$ is the set of all non-empty filters F of $\underline{C}$ and $\mathfrak{I}(\underline{C})$ is the set of all ideals I of $\underline{C}$ with $F \Delta I :\iff F \cap I \neq \emptyset$; hence a filter as 'object' has an ideal as ' attribute' if and only if filter and ideal have at least one idea in common. Important are the *ideal-maximal filters* F in $\mathfrak{F}(\underline{C})$ for which an ideal I exists in $\mathfrak{I}(\underline{C})$ sothat F is a maximal filter having the property $F \cap I = \emptyset$; F is named an *I-maximal filter* and, furthermore, if I is a maximal ideal with $F \cap I = \emptyset$ then I is called an *F-opposite*. As dual notions we have *filter-maximal ideals*, *F-maximal ideals*, and *I-opposites*. The set of all ideal-maximal filters is denoted by $\mathfrak{F}_0(\underline{C})$ and the set of all filter-maximal ideals is denoted by $\mathfrak{I}_0(\underline{C})$. The following theorem informs about meaningful structural properties of the concept lattice of $\mathbb{K}(\underline{C})$ (cf. [Ur78], [SW86], [Ha92]):

Theorem 1. [Wi08] *The ordered set $\underline{C}$ of ideas is naturally embedded by the map $\iota : x \mapsto (\{F \in \mathfrak{F}(\underline{C}) \mid x \in F\}, \{I \in \mathfrak{I}(\underline{C}) \mid x \in I\})$ into the concept lattice of the derived context $\mathbb{K}(\underline{C})$ where $\iota(x \wedge y) = \iota(x) \wedge \iota(y)$ resp. $\iota(x \vee y) = \iota(x) \vee \iota(y)$ if $x \wedge y$ resp. $x \vee y$ exists in $\underline{C}$; in $\underline{\mathfrak{B}}(\mathbb{K}(\underline{C}))$, the set of all $\bigvee$-irreducibles $J(\underline{\mathfrak{B}}(\mathbb{K}(\underline{C})))(= \gamma\mathfrak{F}_0(\underline{C}))$ is $\bigvee$-dense and the set of all $\bigwedge$-irreducibles $M(\underline{\mathfrak{B}}(\mathbb{K}(\underline{C})))(= \mu\mathfrak{I}_0(\underline{C}))$ is $\bigwedge$-dense, i.e., $\underline{\mathfrak{B}}(\mathbb{K}(\underline{C})) \cong \underline{\mathfrak{B}}(\mathfrak{F}_0(\underline{C}), \mathfrak{I}_0(\underline{C}), \Delta)$.*

Proof. For $x \in C$, $(\{F \in \mathfrak{F}(\underline{C}) \mid x \in F\}, \{I \in \mathfrak{I}(\underline{C}) \mid x \in I\})$ is a formal concept of the formal context $\mathbb{K}(\underline{C})$. This can be concluded from $[x) \in \{F \in \mathfrak{F}(\underline{C}) \mid x \in F\} = \{(x]\}^{\Delta}$ and $(x] \in \{I \in \mathfrak{I}(\underline{C}) \mid x \in I\} = \{[x)\}^{\Delta}$. Now, with the equivalences $x \leq y \iff (x] \subseteq (y] \iff \{(x]\}^{\Delta} \supseteq \{(y]\}^{\Delta} \iff \gamma[x) \leq \gamma[y)$, it follows that ι is an order embedding of $\underline{C}$ into $\underline{\mathfrak{B}}(\mathbb{K}(\underline{C}))$. ι is $\wedge$- resp. $\vee$-preserving which immediately follows by the equivalences $x \wedge y \in F \iff x \in F$ and $y \in F$ resp. $x \vee y \in I \iff x \in I$ and $y \in I$.

Now, let F_0 be an ideal-maximal filter of $\underline{C}$ with an F_0-opposite I. Since γF_0 has the extent $\{F \in \mathfrak{F}(\underline{C}) \mid F_0 \subseteq F\}$, $\gamma F_0 \wedge \mu I$ has the extent $\{F \in \mathfrak{F}(\underline{C}) \mid F_0 \subset F\}$. Thus γF_0 is $\bigvee$-irreducible and has $\gamma F_0 \wedge \mu I$ as its unique lower neighbour; this yields $\gamma F_0 \in J(\underline{\mathfrak{B}}(\mathbb{K}(\underline{C})))$ and hence $\gamma\mathfrak{F}_0(\underline{C}) \subseteq J(\underline{\mathfrak{B}}(\mathbb{K}(\underline{C})))$. Conversely, let $\mathfrak{b}$ be a $\bigvee$-irreducible formal concept of $\underline{\mathfrak{B}}(\mathbb{K}(\underline{C}))$. By the Basic Theorem on Concept Lattices $\mathfrak{b}$ must be an object concept, i.e., there is an $F \in \mathfrak{F}(\underline{C})$ with $\mathfrak{b} = \gamma F$. For the unique lower neighbour $\mathfrak{c}$ of $\mathfrak{b}$ there exists, by the Lemma of Zorn, a maximal formal concept $\mathfrak{d}$ with $\mathfrak{c} \leq \mathfrak{d}$ and $\mathfrak{b} \not\leq \mathfrak{d}$. It follows that $\mathfrak{d}$ is

[1] Filters und ideals represent dual processes of convergence of ordered ideas.

$\bigwedge$-irreducible in $\underline{\mathfrak{B}}(\mathbb{K}(\underline{C}))$ and is therefore an attribute concept by the Basic Theorem; hence there is an $I \in \mathfrak{I}(\underline{C})$ with $\mathfrak{d} = \mu I$. Because of $\gamma F \wedge \mu I = \mathfrak{c}$, F is I-maximal and so $F \in \mathfrak{F}_0(\underline{C})$ which proves $\gamma\mathfrak{F}_0(\underline{C}) = J(\underline{\mathfrak{B}}(\mathbb{K}(\underline{C})))$. Dually, we obtain $M(\underline{\mathfrak{B}}(\mathbb{K}(\underline{C}))) = \mu\mathfrak{I}_0(\underline{C})$.

Finally let F be an arbitrary filter of $\underline{C}$. For each element $a \in C \setminus F$ and its principal ideal $(a]$ there is an $(a]$-maximal filter F_a with $F \subseteq F_a$ by the Lemma of Zorn. It follows that $F_a \in \mathfrak{F}_0(\underline{C})$ and $F = \bigcap_{a \in C \setminus F} F_a$ for all $a \in C \setminus F$ and hence $\gamma F = \bigvee_{a \in C \setminus F} \gamma F_a$. Therefore we can conclude with the Basic Theorem that $\gamma\mathfrak{F}_0(\underline{C})$ is $\bigvee$-dense in $\underline{\mathfrak{B}}(\mathbb{K}(\underline{C}))$. Dually we obtain that $\mu\mathfrak{I}_0(\underline{C})$ is $\bigwedge$-dense in $\underline{\mathfrak{B}}(\mathbb{K}(\underline{C}))$. □

5 Conceptual Extension of Convex Continuum Structures

Theorem 1 yields a *general method* to derive from an ordered set of preconceptual ideas a concept lattice in which every formal concept is the supremum of $\bigvee$-irreducible concepts and the infimum of $\bigwedge$-irreducible concepts. For applying Theorem 1 to *two-dimensional convex continuum structures* (introduced in Definition 1), the ideal-maximal filters and filter-maximal ideals of those continuum structures are determined by the following lemma.

Lemma 1. *Let $\underline{CS2}$ be a two-dimensional convex continuum structure. In the ordered set $\underline{CS2}$, the dense chains $F_j := C_j \cup \{\mathbf{1}\}$ $(j \in J)$ are the 'extreme' ideal-maximal filters and their complements $I_j := CS2 \setminus F_j$ $(j \in J)$ are the 'extreme' filter-maximal ideals. Other ideal-maximal filters F_{c_j} of $\underline{CS2}$ are represented by the fitting cuts $\{c_j^{\perp}, c_j^{\top}\}$ of $\underline{CS2}$ $(j \in J)$, each passing through a fixed infinitesimal hole which is determined by adding another fitting cut $\{c_j^{\dashv}, c_j^{\vdash}\}$ through the same hole; further fitting cuts through such a hole, each combined with the underlying cut $\{c_j^{\perp}, c_j^{\top}, \}$, yield four types of subcontinua $c_j^{\rightarrow} \wedge c_j^{\perp} \wedge c_j^{\dashv}$, $c_j^{\leftarrow} \wedge c_j^{\perp} \wedge c_j^{\vdash}$, $c_j^{\nearrow} \wedge c_j^{\top} \wedge c_j^{\dashv}$, and $c_j^{\nwarrow} \wedge c_j^{\top} \wedge c_j^{\vdash}$ (see Fig. 3). The four types of these filters can be defined as follows:*

$$
\begin{aligned}
F_{c_j^{\rightarrow}} &:= \{x \in C_j \mid \exists\, c_j^{\rightarrow} \in C_j^{\rightarrow}, c_j^{\perp} \in C_j^{\perp}, c_j^{\dashv} \in C_j^{\dashv}, d_j^{\vdash} \in D_j^{\vdash} : x \geq c_j^{\rightarrow} \wedge c_j^{\perp} \wedge c_j^{\dashv} \wedge d_j^{\vdash} > o\},\\
F_{c_j^{\leftarrow}} &:= \{x \in C_j \mid \exists\, c_j^{\leftarrow} \in C_j^{\leftarrow}, c_j^{\perp} \in C_j^{\perp}, c_j^{\vdash} \in C_j^{\vdash}, d_j^{\dashv} \in D_j^{\dashv} : x \geq c_j^{\nearrow} \wedge c_j^{\top} \wedge c_j^{\dashv} \wedge d_j^{\vdash} > o\},\\
F_{c_j^{\nearrow}} &:= \{x \in C_j \mid \exists\, c_j^{\nearrow} \in C_j^{\nearrow}, c_j^{\top} \in C_j^{\top}, c_j^{\dashv} \in C_j^{\dashv}, d_j^{\vdash} \in D_j^{\vdash} : x \geq c_j^{\leftarrow} \wedge c_j^{\perp} \wedge c_j^{\vdash} \wedge d_j^{\dashv} > o\},\\
F_{c_j^{\nwarrow}} &:= \{x \in C_j \mid \exists\, c_j^{\nwarrow} \in C_j^{\nwarrow}, c_j^{\top} \in C_j^{\top}, c_j^{\vdash} \in C_j^{\vdash}, d_j^{\dashv} \in D_j^{\dashv} : x \geq c_j^{\nwarrow} \wedge c_j^{\top} \wedge c_j^{\vdash} \wedge d_j^{\dashv} > o\};
\end{aligned}
$$

the corresponding filter-maximal ideals of $\underline{CS2}$ can be defined as follows:

$$
\begin{aligned}
I_{c_j^{\rightarrow}} &:= \{x \in C_j^{\perp} \mid \exists\, \bar{x} \in C_j^{\perp} : x \leq \bar{x} < c_j^{\perp} \setminus c_j^{\rightarrow}\} \text{ and } I_{c_j^{\top}} := \{y \in C_j \mid y \leq c_j^{\top}\},\\
I_{c_j^{\nearrow}} &:= \{x \in C_j^{\top} \mid \exists\, \bar{x} \in C_j^{\top} : x \leq \bar{x} < c_j^{\top} \setminus c_j^{\nearrow}\} \text{ and } I_{c_j^{\perp}} := \{y \in C_j \mid y \leq c_j^{\perp}\},\\
I_{c_j^{\leftarrow}} &:= \{x \in C_j^{\perp} \mid \exists\, \bar{x} \in C_j^{\perp} : x \leq \bar{x} < c_j^{\perp} \setminus c_j^{\leftarrow}\} \text{ and } I_{c_j^{\top}} := \{y \in C_j \mid y \leq c_j^{\top}\},\\
I_{c_j^{\nwarrow}} &:= \{x \in C_j^{\top} \mid \exists\, \bar{x} \in C_j^{\top} : x \leq \bar{x} < c_j^{\top} \setminus c_j^{\nwarrow}\} \text{ and } I_{c_j^{\perp}} := \{y \in C_j \mid y \leq c_j^{\perp}\}.
\end{aligned}
$$

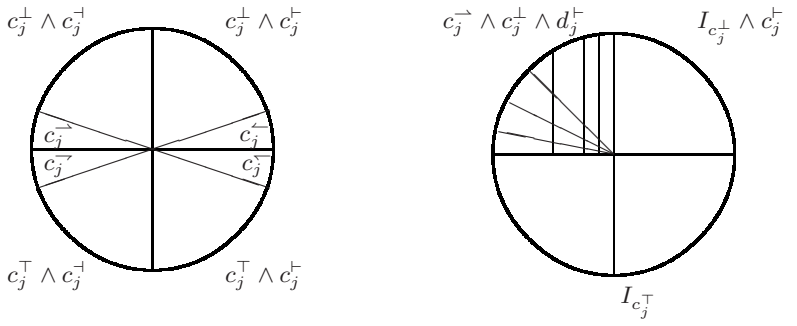


Fig. 3. The diagram indicates how an ideal-maximal filter can be constructed within the two-dimensional convex continuum structure $\underline{CS2}$. Each interior of the two circles represents a two-dimensional convex continuum structure which is divided by cuts into four subcontinua, respectively. In the left diagram two further cuts allow to notify a triangular element of each of the four defined ideal-maximal filters. The right diagram sketches in the upper-left how the filter converges to an infinite hole by using centring cuts and parallel cuts to form smaller and smaller triangles.

Proof. Since $CS2$ is the disjoint union of F_j and I_j for each $j \in J$, F_j is $I_j - maximal$ and I_j is F_j-maximal. For each fitting cut $\{c_j^{\perp}, c_j^{\top}\}$ through a fixed infinitesimal hole, there are four further decompositions of $CS2$ into the disjunct components

1) $F_{c_j^{\rightharpoonup}}, I_{c_j^{\rightharpoonup}}, I_{c_i^{\top}}$, 2) $F_{c_j^{\leftharpoondown}}, I_{c_j^{\leftharpoondown}}, I_{c_i^{\perp}}$, 3) $F_{c_j^{\rightharpoondown}}, I_{c_j^{\rightharpoondown}}, I_{c_i^{\top}}$, 4) $F_{c_j^{\leftharpoonup}}, I_{c_j^{\leftharpoonup}}, I_{c_i^{\perp}}$,

respectively. The first component is always an ideal-maximal filter according to the second and to the third component; conversely, the second and the third component are filter-maximal ideals according to the first component.

Now , let F_* be an ideal-maximal filter in the ordered set $\underline{CS2}$ which is different from the already identified ideal-maximal filters. Let $\{c_j^{\perp}, c_j^{\top}\}$ be a fitting cut of $\underline{CS2}$. Then there exists a filter basis of subcontinua in F_* which belongs completely to one side of the fitting cut. It follows that there are infinitely many disjoint fitting cuts between the chosen fitting cut and the subcontinua of the fixed filter basis. This has as consequence that such a filter basis is completely separated from the fitting cuts and the infinitesimal holes. Therefore, the theory of two-dimensional convex continuum structures shall be restricted to the ideal-maximal filters which are in the boundary of fitting cuts and infinitesimal holes. □

Theorem 2. *In the concept lattice of the formal context* $\mathbb{K}(\underline{CS2}) := (\mathfrak{F}(\underline{CS2}), \mathfrak{I}(\underline{CS2}), \Delta)$ *of a two-dimensional convex continuum structure* $\underline{CS2}$,

(1) $\iota(\mathbf{1})\,(= \gamma F_j \vee \gamma F_k$ *if* $j \neq k)$ *is the greatest element of* $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$,
(2) $\gamma\mathfrak{F}_0(\underline{CS2})$ *is the set of all atoms of* $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$ *which are created by the extreme ideal-maximal filters* F_j *and the quadruples of ideal-maximal filters* $F_{c_j^{\rightharpoonup}}, F_{c_j^{\leftharpoondown}}, F_{c_j^{\rightharpoondown}}, F_{c_j^{\leftharpoonup}}$ *lying on the boundary of an infinitesimal hole and on the boundary of a fitting cut passing through that hole,*

(3) *$\mu\mathfrak{I}_0(\underline{CS2})$ is the set of all $\wedge$-irreducible elements which disintegrates into an infinite collection of dense chains forming together a dense cylinder,*
(4) *the quadruples of atoms on the boundaries of infinitesimal holes which belong to a fitting cut $\{c^{\perp}, c^{\top}\}$ and its associated cut always generate isomorphic $\vee$-semilattices consisting of 9 elements; furthermore, each atom representing an extreme ideal-maximal filter together with a corresponding pair of atoms belonging to $F_{c_j^{\rightarrow}}$, $F_{c_j^{\leftarrow}}$, or $F_{c_j^{\neg}}$, $F_{c_j^{\urcorner}}$ always generate isomorphic $\vee$-semilattices consisting of 4 elements (see Fig. 4),*
(5) *two quadruples of atoms on the boundaries of infinitesimal holes which belong to a fitting cut $\{c^{\perp}, c^{\top}\}$ generate two isomorphic $\vee$-semilattices consisting of 9 elements which combine to a third isomorphic $\vee$-semilattice directly above the two considered quadruples; the third $\vee$-semilattice can also be constructed out of two 4-element $\vee$-semilattices contained in infinitesimal holes belonging to two extreme ideal-maximal filters (see Fig. 5).*

Proof. (1): $I \in \{F_j\}^{\Delta} \cap \{F_k\}^{\Delta}$ (if $j \neq k$) implies $I = (\mathbf{1}] = \mathbf{CS2}$. Therefore $\iota(\mathbf{1}) = \gamma\mathbf{F_j} \vee \gamma\mathbf{F_k}$ and $\iota(\mathbf{1})$ is the greatest element of $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$.
(2): By Theorem 1, $\gamma\mathfrak{F}_0(\underline{CS2})$ is $\bigvee$-dense in $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$ and consists exactly of the $\bigvee$-irreducible formal concepts of $\mathbb{K}(\underline{CS2})$. Hence, by Lemma 1, the formal concepts in $\gamma\mathfrak{F}_0(\underline{CS2})$ are exactly the atoms of $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$, which are constructed by the extreme ideal-maximal filters and by the ideal-maximal filters collected as quadruples in the derived infinitesimal holes.
(3): By Theorem 1, $\mu\mathfrak{I}_0(\underline{CS2})$ is $\bigwedge$-irreducible in $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$ and consists exactly of the $\bigwedge$-irreducible formal concepts of $\mathbb{K}(\underline{CS2})$. By Lemma 1, each of these $\bigwedge$-irreducible concepts lies in one of the infinitely mainy dense chains $[\gamma F_j, \iota(\mathbf{1})[$. Since $\mathfrak{b} = \bigwedge\{\mu I \in [\gamma F_j, \iota(1)[\mid\mid \mu I \geq \mathfrak{b}\} \vee \bigwedge\{\mu[\gamma F_k, \iota(\mathbf{1})[\mid\mid \mu\mathbf{I} \geq \mathfrak{b}\}$, $[\gamma F_j, \iota(\mathbf{1})[\cup [\gamma\mathbf{F_k}, \iota(\mathbf{1})[$ is the set of all $\wedge$-irreducible formal concepts of $\mathbb{K}(\underline{CS2})$ $(j \neq k)$.
(4): A quadruple of atoms in an infinitesimal hole determined by a fitting cut and an associated cut corresponds to the quadruple of disjoint subcontinua created by the chosen fitting cut and its associated cut. The suprema of those subcontinua obviously represent the 9-element $\vee$-semilattice generated by the considered quadruple of atoms. Two atoms, which correspond to two adjacent subcontinua, and the atom at the end of the fitting cut between the two adjacent subcontinua, which represents an extreme ideal-maximal filter, lead to the 4-element $\vee-$lattice.
(5): The pairs of 4-element $\vee$-lattices are isomorphic to pairs of $\vee$-sublattices the atoms of which correspond with the atoms of the considered 9-element $\vee$-sublattice. □

Since the two-dimensional convex continuum structure $\underline{CS2}$ is *embeddable* into the concept lattice $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$ by Theorem 1, that concept lattice yields for the two-dimensional convex continuum structure an extended conceptual coherence which is made explicit in several aspects by Theorem 2. Most important for the theme of this paper is that the atoms of $\underline{\mathfrak{B}}(\mathbb{K}(\underline{CS2}))$ represent '*point*

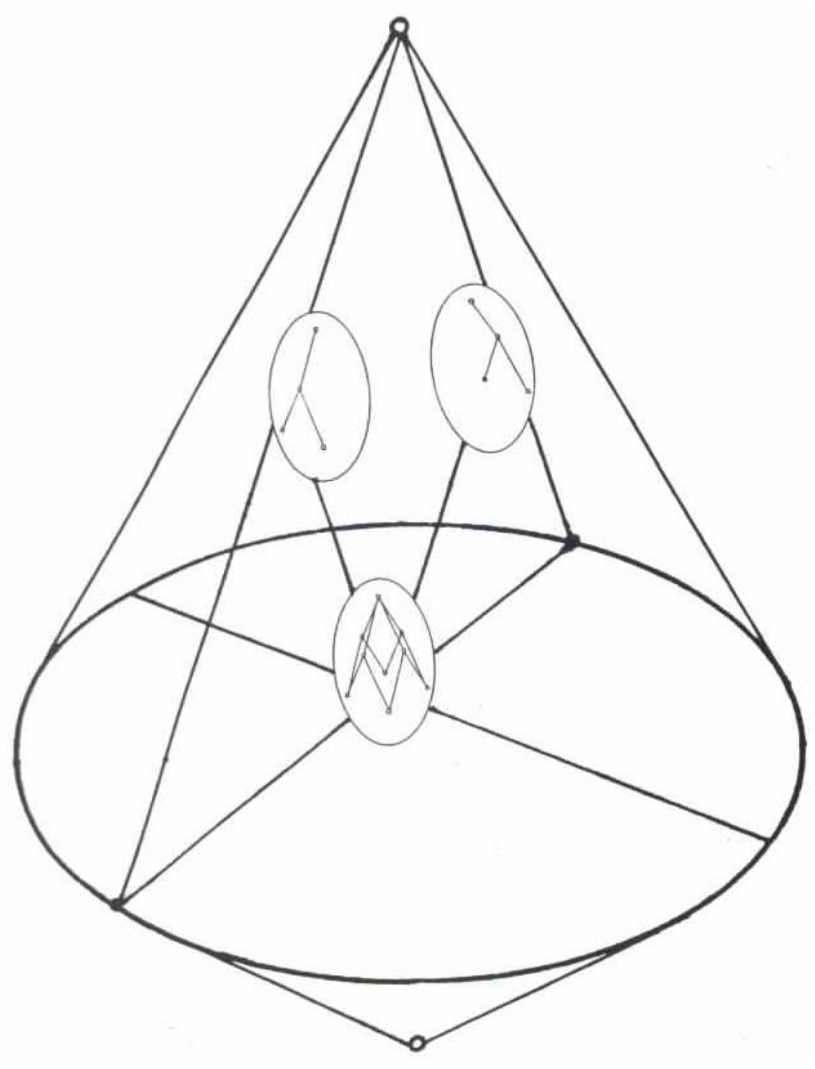


Fig. 4. The diagram above sketches a concept lattice $(\mathfrak{F}(\underline{CS2}), \mathfrak{I}(\underline{CS2}), \Delta)$ by visualizing how a subsemilattice is generated by a quadruple of ideal-maximal filters $F_{c_j^{\rightharpoonup}}$, $F_{c_j^{\leftharpoonup}}$, $F_{c_j^{\rightharpoondown}}$, $F_{c_j^{\leftharpoondown}}$ which is generated as the infimum of the two four-element subsemilattices of the corresponding $\wedge$-irreducible boundary

concepts', which have each other formal concept as supremum by (2). By (4), each fitting cut of the two-dimensional convex continuum structure gives rise to four cut-limiting point concepts the supremum of which may be viewed as a point in the common sense. (5) describes how point concepts can be represented as limits of continua. These hints shall suffice to demonstrate the fruitfulness of the concept-analytic method and support Aristotle's conception of continua.

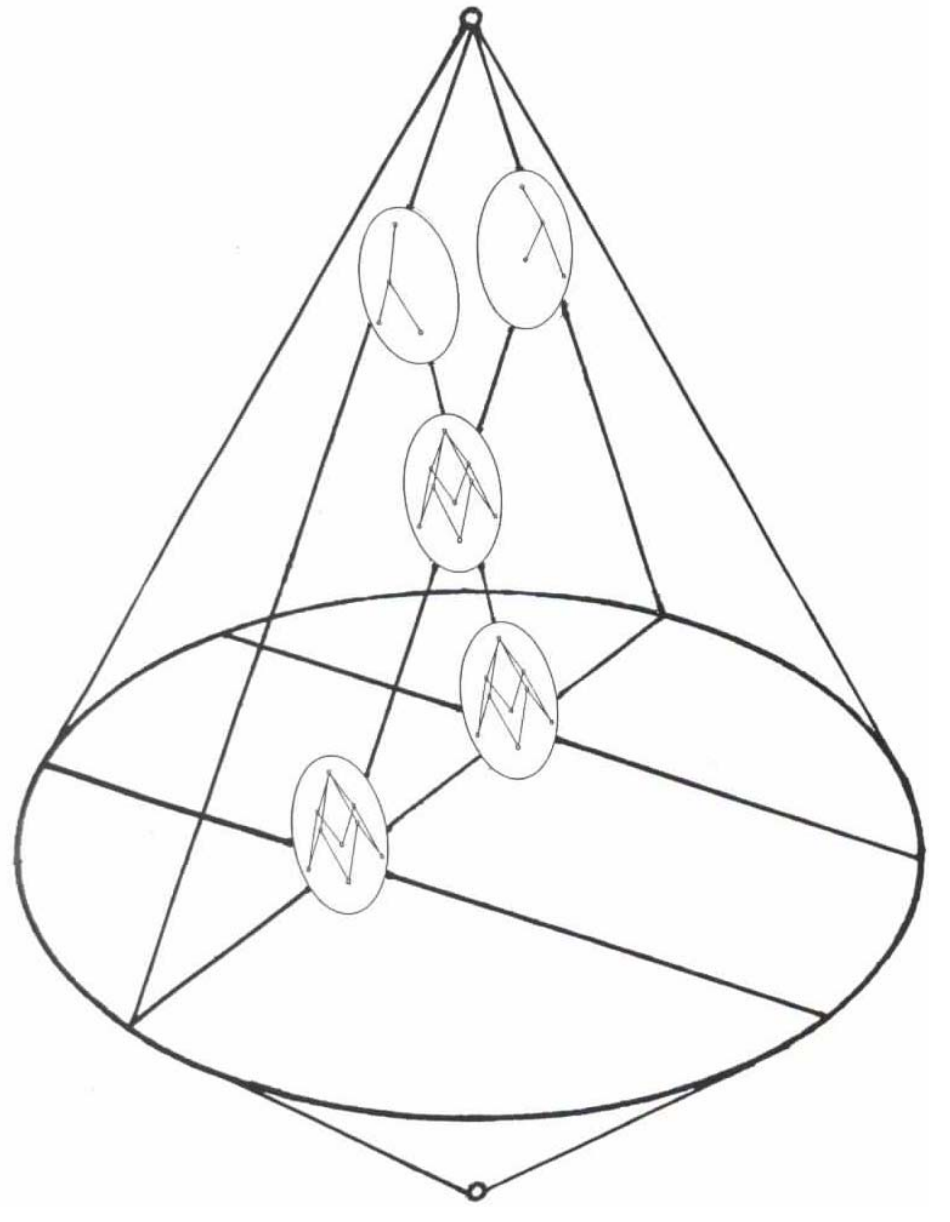


Fig. 5. The diagram above sketches a concept lattice $(\mathfrak{F}(\underline{CS2}), \mathfrak{I}(\underline{CS2}), \Delta)$ by visualizing how a subsemilattice is generated by two quadruples of ideal-maximal filters $F_{c_j^{\rightarrow}}$, $F_{c_j^{\leftarrow}}$, $F_{c_j^{\rightarrow}}$, $F_{c_j^{\leftarrow}}$ and $F_{d_j^{\rightarrow}}$, $F_{d_j^{\leftarrow}}$, $F_{d_j^{\rightarrow}}$, $F_{d_j^{\leftarrow}}$ which is also generated as the infimum of the two four-element subsemilattices of the corresponding $\wedge$-irreducible boundary

References

[Ar95] Aristoteles Werke in deutscher Übersetzung. Bd. 11: Physikvorlesung. 5. Aufl. Akademie Verlag, Berlin (1995)

[GW99] Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)

[Ha92] Hartung, G.: A topological representation of lattices. Algebra Universalis 29, 273–299 (1992)

[La86] Laugwitz, D.: Zahlen und Kontinuum. B.I.-Wissenschaftsverlag, Mannheim (1986)

[Pi59] Piaget, J.: La formation du symbole chez l'enfant-imitation, jeu et rêve - Image et représentation. Delachaux et Niestlé S.A., Neuchâtel (1959)

[SW86] Stahl, J., Wille, R.: Preconcepts and set representations of contexts. In: Gaul, W., Schader, M. (eds.) Classification as a tool of research, pp. 431–438. North-Holland, Amsterdam (1986)

[Ur78] Urquhart, A.: A topological representation theory for lattices. Algebra Universalis 8, 45–58 (1978)

[Wi82] Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered sets, pp. 445–470. Reidel, Dordrecht (1982)

[Wi08] Wille, R.: Formal Concept Analysis of one-dimensional continuum structures. Algebra Universalis 59, 197–208 (2008)

Counting of Moore Families for n=7

Pierre Colomb[1], Alexis Irlande[2], and Olivier Raynaud[1]

[1] Université Blaise Pascal, Campus Universitaire des Cézeaux, 63173 Aubière, France
[2] Universidad Nacional de Colombia, Bogota, Colombia

Abstract. Given a set $U_n = \{0, 1, ..., n-1\}$, a collection $\mathcal{M}$ of subsets of U_n that is closed under intersection and contains U_n is known as a Moore family. The set of Moore families for a given n, denoted by $\mathtt{M}_n$, increases very quickly with n, thus $|\mathtt{M}_3|$ is 61 and $|\mathtt{M}_4|$ is 2480. In [1] the authors determined the number for $n = 6$ and stated a 24h- computation-time. Thus, the number for $n = 7$ can be considered as an extremely difficult technical challenge. In this paper, we introduce a counting strategy for determining the number of Moore families for $n = 7$ and we give the exact value : 14 087 648 235 707 352 472. Our calculation is particularly based on the enumeration of Moore families up to an isomorphism for n ranging from 1 to 6.

1 Introduction

The counting (and/or enumeration) of a large set of mathematical objects is a pleasant challenge. This kind of exercise requires original algorithmic processes, which involves a thorough knowledge of the properties of the objects to be counted, efficient search data structures, but also, and above all, state-of-the-art programming techniques. In this paper, our efforts have been concentrated on the counting of Moore families generated by a given set $U_n = \{0, 1, ..., n-1\}$. The concept of Moore family, or of closure operator (extensive, isotone and idempotent function of 2^{U_n} in 2^{U_n}), or of implicational system, is applied in numerous fields. For example, let's consider mathematics research such as [2] for algebra, computer science such as [3] for the theory of orders and lattices, [4] for relational databases and finally [5] and [6] for data analysis. The name 'Moore family' was first used by Birkhoff in [7] referring to E.H. Moore 's early century research in [8]. Technically, a Moore family on U_n, denoted by $\mathcal{M}$, is a collection of sets (or family) closed under intersection and containing U_n (cf. figure 1).

The set of Moore families on U_n, denoted $\mathtt{M}_n$, is itself a closure system (a closure system being the set of the fixed points of a closure operator). Thus, the system composed of Moore families contains one maximum element (2^{U_n} : all subsets of U_n) and the intersection of two Moore families is a Moore family itself. To get an overall view of the properties of this closure system, see [9]. Ordering the set of elements of a closure system by inclusion, we get a lattice structure. Indeed, an inclusion ordering of the set of elements of a closure system can generate a lattice structure.

In [10], Burosch considers the issue of counting Moore families as natural, so he suggests an upper bound for that number. In this paper, we will complete

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 72–87, 2010.

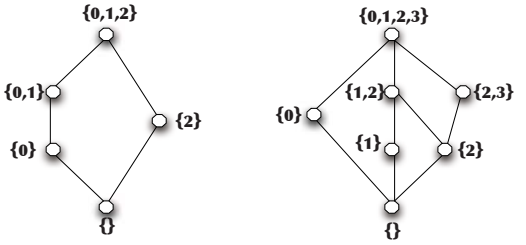


Fig. 1. Two moore families : On the left, family $\{\emptyset, \{0\}, \{0,1\}, \{2\}, \{0,1,2\}\}$ on the set $\{0,1,2\}$. On the right, family $\{\emptyset, \{0\}, \{1\}, \{2\}, \{1,2\}, \{2,3\}, \{0,1,2,3\}\}$ on the set $\{0,1,2,3\}$.

his survey using a new $|\mathtt{M}_n|$-bound based on $|\mathtt{M}_{n-1}|$. In 1998, Higuchi in [11] calculated $|\mathtt{M}_5|$ by a depth first search study of the covering graph of a Moore family lattice structure. More recently, Habib and Nourine have evaluated in [1] the size of $\mathtt{M}_6$. Their method is based on the existence of a bijection between the set of Moore families and the ideals colorset of a colored poset composed of boolean lattices. Thanks to an efficient algorithm of enumeration of order ideal sets (cf. [12]), the authors managed to count $\mathtt{M}_6$ and stated that the process would take 24h on a pentium III 600 MegaHertz (we note that the method presented here compute this number $\mathtt{M}_6$ in around 120ms on a Core2quad Q9300 2,5 GigaHertz). The whole set of values of this counting is given in table 1. Considering the exponential development of the results, the evaluation of the number for $n = 7$ turned out to be a particularly difficult challenge.

Table 1. Known values of $|\mathcal{M}_n|$ on $n \leq 7$

n	$\lvert\mathtt{M}_n\rvert$	Référence
0	1	
1	2	
2	7	
3	61	
4	2 480	
5	1 385 552	[11]
6	75 973 751 474	[1]
7	14 087 648 235 707 352 472	This paper

The rest of the paper is composed as follows. The second part is devoted to the data structure and key points on which our calculation strategy was based: **symmetry concept**, **canonical form** and **maximal family**. In the third part, we will present the main algorithmic principles we have implemented. Then, in the fourth part, we will deal with the technological aspects of the calculation process (type of machine, performance, reliability index). As a conclusion, we will put things into perspective.

2 Strategy Elements

This part defines the essential elements of our calculation strategy. Firstly, we will describe the coding used to store and process the Moore families. Then, we will explain the various concepts used as the basis of this calculation strategy: The symmetries between the families, the concept of canonical form as the identifier of an equivalence class as well as the concept of maximal family based on the recursive structure of the objects to be counted. The proofs of the propositions are given in the appendix.

In the introduction, we have defined a Moore family on U_n as a collection of sets containing U_n and closed by intersection. However for the reasons of convenience that the reader will find in the course of reading, our entire algorithmic process will concentrate on enumeration of families closed by union and containing the empty set. All the Moore families are in fact in bijection with this set. Actually, for a family closed by union containing the empty set, one only has to complement every set to obtain a Moore family (and vice-versa).

For example, the U_3 family $\{\{0\},\{0,1\},\{0,2\},\{0,1,2\}\}$ corresponds to the Moore family $\{\emptyset,\{1\},\{2\},\{1,2\}\}$ and vice versa.

2.1 Encoding

Let us consider a $U_n = \{0,...,n-1\}$ universe with n elements and a set $E \subseteq U_n$. E can be naturally encoded by its characteristic vector (for example, refer to figure 2).

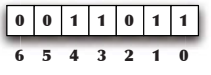


Fig. 2. For the universe $U_7 = \{0,...,6\}$, the set $\{0,1,3,4\}$ is encoded by a 7 bit vector $b[]$ such that $b[i] = 1$ if and only if $i \in \{0,1,3,4\}$

We can associate a decimal value to each subset of E by interpreting the characteristic vector as a binary number. To be more precise, an integer between 0 and 2^n-1 corresponds to each sub-set of $\{0,...,n-1\}$. In the previous example, the integer associated to the set $\{0,1,3,4\}$ is equal to the sum of $2^0+2^1+2^3+2^4$ i.e. 27. By using these decimal values as set identifiers, we can also encode a set family on a universe with n elements by a characteristic vector of 2^n bits (for example, refer to figure 3).

Finally, we associate an integer between 0 and 2^{2^n} with each family by interpreting the new vector as a binary number. For example, 131 is the identifier of the family $\{\emptyset,\{0\},\{0,1,2\}\}$. As we will see, using such an encoding enables carrying out normal operations on the sets by simple logic or arithmetic operations.

Two operations are essential in our counting process:

Testing whether a family contains the empty set and testing whether after adding an E set to an $\mathcal{F}$ family closed by union, $\mathcal{F}$ remains closed by union. The

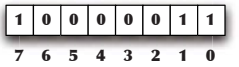


Fig. 3. Let be a universe $U_3 = \{0, 1, 2\}$, the family $\mathcal{F} = \{\emptyset, \{0\}, \{0, 1, 2\}\}$ is encoded by the $2^3 = 8$ bits vector $b[]$ where $b[i] = 1$ if and only if the set E belongs to $\mathcal{F}$ (i being the integer associated with the set E)

first of the two tests becomes trivial when the proposed coding is used. In fact, since the empty set is borne by the weakest bit, a family contains the empty set if and only if the identifier of this family is an odd number. The second, more difficult test requires verifying whether each element of $\mathcal{F}$, the union of this element with E is in $\mathcal{F}$. Nevertheless, this test can be carried out by extracting the following code:

```
for (i = 1 ; i < N ; i++ )
      if ( ( F \& (1U<<i)) \&\& ! ( F \& (1U << (E | i) ) ) )
       printf("F Union E is not union closed");
```

The *for* loop scans all the characteristic integers i of a set. The first part of the test $if\ (\mathcal{F}\&(1U << i))$ verifies whether the set corresponding to the integer i belongs to the $\mathcal{F}$ family. For this test, we create a vector made up of 0 except of 1 at the position i using the $(1U << i)$ expression and we carry out an *and* logic with the $\mathcal{F}$ family. This test naturally gives "true" as result if and only if the i set belongs to $\mathcal{F}$. The second part of the test verifies whether $i \cup E$ belongs to the $\mathcal{F}$ family. Since the identifier of the $E \cup i$ set corresponds to the decimal value of $E|i$, it is enough to carry out an *and* logic between $\mathcal{F}$ and a vector made up of 0 except 1 at the $E|i$ position.

2.2 Symmetry and Canonical Form

A permutation Φ on a finite set $U_n = \{0, ..., n-1\}$ is a bijective function from U_n to U_n. For convenience Φ is often represented by the sequence of its images $\Phi(0), \Phi(1), ..., \Phi(n-1)$. All the permutations on a U_n set are marked by Sym_n. For a set $E \subseteq U_n$ and $\Phi \in Sym_n$, we use $\Phi(E)$ to mark the image of E by Φ defined by :$\Phi(E) = \{\Phi(x)|x \in E\}$. Similarly, we simply use $\Phi(\mathcal{F}) = \{\Phi(E)|E \in \mathcal{F}\}$ for all $\mathcal{F} \subseteq 2^{U_n}$.

Example: Let $\mathcal{F}$ be the family $\{\emptyset, \{2\}, \{1, 2\}, \{0, 1, 2\}\}$ and Φ the permutation $1, 2, 0$ then $\Phi(\mathcal{F}) = \{\emptyset, \{0\}, \{0, 2\}, \{0, 1, 2\}\}$.

Using the concept of permutation we can divide all the families on U_n into equivalence classes. Thus we can say that two families belong to the same class if they are the images of one another by a Sym_n permutation. The reference family of each class is called the **canonical form**. We have used an identification of the canonical form based on the properties of our encoding. Hence, the canonical

Table 2. For $\mathcal{F} = \{\emptyset, \{2\}, \{1,2\}, \{0,1,2\}\}$ and each permutation Φ of Sym_3, the image of $\mathcal{F}$ by Φ and its identifier

$Permutation \Phi$	$\Phi(\mathcal{F})$	Identifier
012	$\{\emptyset, \{2\}, \{12\}, \{012\}\}$	209
021	$\{\emptyset, \{1\}, \{12\}, \{012\}\}$	197
102	$\{\emptyset, \{2\}, \{02\}, \{012\}\}$	177
120	$\{\emptyset, \{0\}, \{02\}, \{012\}\}$	163
210	$\{\emptyset, \{0\}, \{01\}, \{012\}\}$	139
201	$\{\emptyset, \{1\}, \{01\}, \{012\}\}$	141

Table 3. The number of Moore families up to an isomorphism and the average size of classes for each value of n. The average size of classes increases drastically with n (and tends to be close to $n!$) which proves that the set of all the Moore families on a universe contains a large number of isomorphic objects.

n	Number of Moore families up to isomorphism	Average size of classes
1	2	1
2	5	1, 40
3	19	3, 21
4	184	13, 48
5	14664	94, 49
6	108 295 846	701, 54

form of a family $\mathcal{F} \subseteq 2^{U_n}$ is defined as the image by one of the permutations of Sym_n having the smallest identifier.

Example: Let us consider $\mathcal{F} = \{\emptyset, \{2\}, \{1,2\}, \{0,1,2\}\}$ family on U_3 as well as the Sym_3 set containing 6 permutations. As per table 2, the canonical form of $\mathcal{F}$ is the $\{\emptyset, \{0\}, \{01\}, \{012\}\}$ family whose identifier is equal to 139.

Proposition 1. *Let $\mathcal{M}$ be a Moore family on U_n and $\Phi \in Sym_n$ be a permutation, then $\Phi(\mathcal{M})$ is a Moore family on U_n.*

The partitioning that exists on the set of families on U_n also pertains to the set of Moore families. Actually, as per the previous property, the image of a Moore family by a permutation remains a Moore family. Therefore, it is possible to enumerate the Moore families by enumerating the representative of each equivalence class and then by counting its images with the help of different permutations. In general, the enumeration of a single representative per equivalence class is called as "enumeration of a set of combination objects up to an isomorphism". This strategy is often based on the observation that a large part of the combinatorial explosion related to all the objects studied can be explained by the presence of isomorphic objects. Table 3 shows that this situation is specially verified for the set of Moore families. Let's note that it's common to use symmetry to decrease combinatorial explosion (see for example [13,14]).

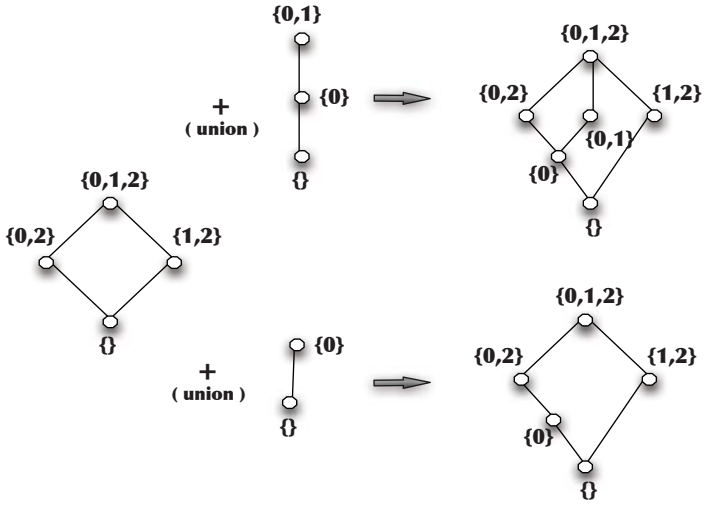


Fig. 4. On left a Moore family on U_3 (such that each element contains 2) associated to two different Moore families on U_{n-1}. In each case, the obtained family, on right, is itself a Moore family.

2.3 Maximal Family

A Moore family $\mathcal{M}$ on U_n can be divided into two parts. The part made up of $\mathcal{M}$ sets containing the $\{n-1\}$ element (marked as $\mathcal{M}_{sup}$ for the upper part) and the additional part (marked as $\mathcal{M}_{inf}$ for the lower part). The $\emptyset$ element is duplicated in order to be included in both parts. Naturally $\mathcal{M} = \mathcal{M}_{sup} \cup \mathcal{M}_{inf}$. Besides, $\mathcal{M}_{sup}$ and $\mathcal{M}_{inf}$ are Moore families. The example in figure 4 proves that there can be many compatible lower parts for a fixed upper part (i.e., their combination gives a Moore family).

There is a unique **maximal family** for a given upper $\mathcal{M}_{sup}$ family as all the families compatible with $\mathcal{M}_{sup}$ are the sub-families of the maximal family. To be more specific:

Proposition 2. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n with $n-1 \in M$ for all $M \in \mathcal{M}_{sup} \setminus \{\emptyset\}$. Then there exists a* ***unique Moore family*** *$\mathcal{M}_{max}$ on U_{n-1} compatible with $\mathcal{M}_{sup}$ such that all Moore families on U_n, whose the restriction to its elements containing $n-1$ corresponds to $\mathcal{M}_{sup}$, can be written $\mathcal{M}_{sup} \cup \mathcal{M}_{inf}$, with $\mathcal{M}_{inf} \subseteq \mathcal{M}_{max}$.*

We can make two remarks : First, The maximal family can be defined as $\mathcal{M}_{max} = \{M \in 2^{U_{n-1}} \mid M \cup M' \text{ for all } M' \in \mathcal{M}\}$. Second the maximal family correspond to the set of the quasi-closed sets that don't contain $\{n-1\}$.

The maximal family associated to the $\mathcal{M}_{sup}$ family given in figure 4, for example, is a family made up of $\{\emptyset, \{0\}, \{1\}, \{0,1\}\}$ elements. We can verify whether the two compatible families given in figure 4 are the sub-families of this family.

Algorithm 1. *maximalFamily*()

```
Data   : V[2^n] which corresponds to the family M_sup
Result : V[2^n] which corresponds to the family M_sup ∪ M_max
begin
    for (b = 2^n − 1 to 0) do
        V[b] ← 1;
        if V[] does not corresponds to a union closed family then
            V[b] ← 0;
    return V[];
end
```

Remember, the encoding taken from a Moore family takes the form of a 2^n bit vector (we will use a 128 size vector to count the Moore families on U_7). Naturally the first 2^{n-1} bits encode for all the sets containing the $n-1$ element (i.e. $\mathcal{M}_{sup}$), whereas the last 2^{n-1} bits encode for $\mathcal{M}_{inf}$. Therefore, our counting strategy consists in generating only the upper parts of the vector, and determining the maximal family that is compatible with each of these upper parts (which means calculating the lower part of the vector). Algorithm 1 is a calculation process of this unique family.

Proposition 3. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n (all its elements containing $n-1$), the algorithm 1 computes the maximal family $\mathcal{M}_{max}$ compatible with $\mathcal{M}_{sup}$ and returns the vector which corresponds to the family $\mathcal{M}_{sup} \cup \mathcal{M}_{max}$.*

There are multiple uses of the maximal family concept: Firstly, we have seen that the counting of Moore families of U_7 requires using a 128 bits vector. Unfortunately, integers on 128 bits are not convenient enough to handle directly. Using maximal families enables dividing the calculation of Moore families into two distinct and independent parts by using only the 64 bits vectors. Secondly, it also enables reusing the calculations made while counting the Moore families on U_{n-1} to count the Moore families on U_n. Finally, it helps to highlight a natural bound on the number of Moore families on U_n according to this number on U_{n-1} (refer to proposition 4).

Proposition 4. *Let $\mathtt{M}_n$ and $\mathtt{M}_{n+1}$ be the sets of all Moore family respectively on U_n and U_{n+1} then we have $\mid \mathtt{M}_{n+1} \mid \leq 2 * \mid \mathtt{M}_n \mid^2$.*

The following section concentrates on the implementation of these concepts in an algorithmic framework.

3 Algorithms

This section describes three different algorithms of counting the Moore families on U_n. The first algorithm is a naïve recursive algorithm that scans a tree representing the set of Moore families. The second algorithm introduces the concept

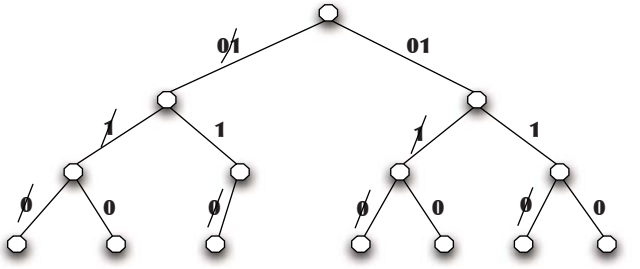


Fig. 5. Binary tree corresponding to the exploration of algorithm 2

Algorithm 2. $Moore_1()$

```
Data  : V[]: bit vector; k :integer
begin
    if k = 0 then
        |M_n| ← |M_n| + 1;
    else
        Moore_1(V[], k − 1);
        V[k] ← 1;
        if V[] corresponds to an union closed family then
            Moore_1(V[], k − 1)
        V[k] ← 0;
end
```

of symmetry by storing the unique representative of each class in a hash table. A "coefficient" variable associated to each class records the size of the class. Lastly, the third algorithm integrates the concept of maximal family in the symmetry concept. Note that only the last algorithm gives reasonable calculation time for $n = 7$.

3.1 Naïve Algorithm

The algorithm 2 completes a recursive scan of the set of Moore families on U_n. The medium for this scan is a tree having Moore families as leaves. It stops when a given set is present or absent in the family. The tree structure of Moore families on U_2 is given in figure 5. Thus the algorithm generates a new set at every node and determines whether it can be added to the current family. If the answer is "No", the process continues on the left branch with the same family. Otherwise, the process is restarted on the right branch with a family integrating the new element. When a leaf is reached, the value of a global variable ($|\mathtt{M}_n|$), counting the total number of families, increases.

Technically, the current family is stored in the $V[]$ vector (initialised to 0) which is updated from left to right. k corresponds to the position of the bit of

$V[]$, i.e. to the set which is likely to be inserted in the family. k initialised to 2^n-1 decreases naturally till 0 which corresponds to a leave of the tree. In this way, the sets are considered in a linear order whose first element is the $\{0, 1, ..., n-1\}$ set and the last element is the empty set.

3.2 Algorithm Using the Symmetries

The algorithmic principle given here integrates the concept of symmetries and stores the canonical forms in a hash table by associating a *coefficient* to each of these forms. Inserting a canonical form coupled with a c coefficient in the table involves creating a new input if the form is not known or increasing the coefficient of the existing form by c.

To calculate the canonical form of the Moore family $\mathcal{M}$, the function $canonicalForm(\mathcal{M})$ generates $\mathcal{M}$ images by all the permutations of Sym_{U_n} and returns the image of the smallest identifier. These small values of n (at the most 7) provide reasonable calculation time for a complexity of $O(n!)$. Finally we can notice that the problem of the canonical form calculation comes down to the problem of graph isomorphism.

Our strategy can be represented using two algorithms:
The first $fillVector()$ (refer to algorithm 3) enables assigning bits to the $V[]$ vector between the k and m positions in such a way that$V[]$ becomes a family closed by union. It is an adaptation of the $Moore_1$ algorithm for which the stop bit (corresponding to a leave) can be parameterised. The second algorithm (refer to algorithm 4) uses the information stored in the hash table at each step to effectively construct the new vectors which can correspond to the Moore families. At the end of the calculation, it is enough to only add the coefficients of each canonical form present in the latest table to find the total number of families.

Let us note that the c coefficient associated to a $V[]$ vector when it is inserted in the hash table is the coefficient associated to the vector which was used when the $fillVector()$ algorithm was called in $Moore_2$ process.

Algorithm 3. $fillVector()$

```
Data  : V[] : bit vector; k,m : integers;
begin
    if k = m − 1 then
        myHashTable.add(canonicalForm(V[]));
    else
        fillVector(V[], k − 1);
        V[k] ← 1;
        if V[] corresponds to an union closed family then
            fillVector(V[], k − 1);
        V[k] ← 0;
end
```

Table 4. Sizes of the hashtable containing Moore families corresponding to different values of n before and after reduction

n	$\|H\|$ before reduction	$\|H\|$ after reduction
2	7	5
3	46	19
4	916	184
5	140 463	14 664
6	770 413 085	108 295 846

Algorithm 4. $moore_2()$

```
Data   : n : integer;
Result : integer / number of Moore families on U_n;
begin
    H'.add(V[] =< 0, 0, ..., 0, 0, 1 >);
    H ← ∅;
    i ← 1;
    while i ≤ n do
        for V[] ∈ H' do
            fillVector(V[], 2^i − 1, 2^(i−1));
        i++;
        H' ← H;
    return Σ_{V[]∈H} V[].coefficient;
end
```

Table 4 recapitulates the size of the table containing the families generated for each value of n. The size is given before and after the reduction. Naturally, we can find the number of Moore families up to an isomorphism in the last column. Finally, let us remember that the hash table before reduction is never constructed since the reduction takes place with the help of the $canonicalForm()$ function as the process progresses.

3.3 Algorithm Using the Maximal Families

The algorithm shown here integrates the principle of maximal family (refer to the previous section) with the concept of symmetry that we have just implemented. At first, an adaptation of the previous algorithm is used to generate the vectors representing the $\mathcal{M}_{sup}$ families up to an isomorphism (the number of families isomorphic to each of these families is stored in the c coefficient). Secondly, we determine the associated $\mathcal{M}_{inf}$ maximal family for each $\mathcal{M}_{sup}$ family. Let us note that a $\mathcal{M}_{inf}$ family is a Moore family on U_{n-1} and its weight p corresponds to the number of Moore families included in this family. *Hence, the counting of the Moore families on U_n corresponds to the sum of the product between the weights and the coefficient of each selected family.*

Algorithm 5. $leftPart()$

```
Data   : n : integer;
Result : H;
begin
    H'.add(V =< 0, 0, ..., 0, 0, 1 >);
    H ← ∅;
    i ← 1;
    while i ≤ n − 1 do
        for V ∈ H' do
            FillVector(V, 2^i − 1 + 2^(n−1), 2^(i−1) + 2^(n−1));
        n++;
        H' ← H;
    return H;
end
```

The input of the $maximalFamily()$ function is a $V[]$ vector of 2^n bits that represents a Moore family on U_n as all its elements contain the $n-1$ element. Therefore, all the bits of $V[]$ included between 1 and $2^{n-1}-1$ are positioned at 0. The function returns a vector of 2^{n-1} bits representing the Moore family on U_{n-1} that corresponds to the maximal family associated to the considered Moore family.

The $leftPart()$ algorithm (refer to algorithm 5) generates vectors up to an isomorphism corresponding to the Moore families on U_n by considering only the $2^{U_n} \setminus 2^{U_{n-1}} \setminus \{n-1\}$ elements. The $Moore2$ algorithm given earlier is adapted considering only the bits included between $2^n - 1$ and $2^{n-1} + 1$.

Algorithm 6. $moore_3()$

```
Data   : n : integer; M_{n−1};
Result : |M_n| the number of Moore family on U_n;
begin
    |M_n| ← 0;
    H ← leftPart(n);
    for V[] ∈ H do
        |M_n| ← |M_n| + canonicalForm(maximalFamily(V[])).weight *
        V[].coefficient;
        V[2^(n−1)] ← 1;
        |M_n| ← |M_n| + canonicalForm(maximalFamily(V[])).weight *
        V[].coefficient;
    retourner |M_n|;
end
```

The $Moore3()$ algorithm (refer to algorithm 6) starts by generating the left parts of the vectors by using the $leftPart()$ algorithm. Two Moore families are associated to each left part: one family containing the $n-1$ element and another

not containing this element. Then the algorithm associates a maximal family to each Moore family. The number of Moore families on U_n are calculated by accumulating the results of the coefficients of the $\mathcal{M}_{sup}$ families by the weights of corresponding $\mathcal{M}_{inf}$ families as the process progresses. The algorithm uses a $\mathcal{M}_{n-1}$ dictionary containing the weights of Moore families on U_{n-1}. The weight of a family can be calculated with the help of the *computeWeight*() algorithm (refer to algorithm 7) which functions conversely to the $fillVector()$ algorithm.

Algorithm 7. *computeWeight*()

```
Data   : V[] : vector, k : integer;
Result : p;
begin
    if k = 2^n then
        p ← p + 1;
    else
        computeWeight(V[], k + 1);
        V[k] ← 0;
        if V[] corresponds to a union closed family then
            computeWeight(V[], k + 1)
end
```

In case of the calculation of number of Moore families on U_7, the number of left parts up to an isomorphism is 108295846 (it corresponds to the number of Moore families on U_6 up to an isomorphism). We find 118540742 maximal families and 108295846 maximal families up to an isomorphism. In other words, each Moore family on U_6 is, at least one time, a maximal family of a family on U_7.

4 Technical Aspects

4.1 Implementation

This program is written in C, compiled with Gcc version 4 for Linux 64 bits. We have used the OpenMP parallel programming library. The main calculation lasted 11 hours on a 2.5 GHz Core2 Q9300. The verification lasted 9 hours on a 1.86 GHz double Xeon E5320 (University of Bogotá). The calculation was simultaneously done on a 2.3 GHz Opteron 8356 quadruplet (LIMOS).

4.2 Reliability

The question of reliability is of primordial importance for any result involving a large number of computer calculations. In this case, the authors would like to highlight the following points:

- The result complies with the previous estimations of one of the authors (between $1, 2.10^{19}$ and $1, 7.10^{19}$) based on an extrapolation of data for $n \leq 6$.
- There is no specific code at $n = 7$ and no special threshold between $n = 6$ and $n = 7$ (for example, size of integers used)
- The same program gives good results for n ranging from 1 to 6 (number of Moore families and the families reduced by isomorphism).
- The code was executed on numerous occasions on different architectures (Intel and AMD).
- The program was compiled with three stable compilers (Gcc v. 4.1, 4.2 and 4.3) and various options on different operating systems (Ubuntu and CentOS).
- The result remained stable at each modification of the size available for the hash tables. Let us note that the modification in the table size badly disrupts the time required for the calculation to complete."
- The result remains unchanged by replacing the canonical form based on the smallest representative of the equivalence class with a canonical form based on a larger representative.

5 Conclusion

The problem of enumerating the Moore families in the n order is a complex issue for which there is no known formula. Even the absence of such a formula has not been proved. Numerous combinatory problems fall in the same case. For example, the number of monotonous Boolean functions known as the Dedekind number. An often supported approach to comprehend such formulae involves counting the number of objects for the first values of n using a systematic procedure. We can find such integer sequences on the well-known On-line Encyclopaedia of Integer Sequences. Our work thus had a dual-objective: Not only enriching the already-known sequence for the number of Moore families, but also highlighting the new properties of the set of Moore families.

In this article, we have proved that we can calculate the search objects without enumerating all of them. Therefore, there is a smaller frame of objects (the Moore families on U_n up to an isomorphism in which all the sets include $n - 1$) from which the total number can be deduced. In other words, for each object of the frame it is sufficient to calculate its associated maximal family (a Moore family on U_{n-1}) and calculate its weight (i.e. the number of Moore families included). This enumeration is carried out only once even if the maximal family is found many times. Since the result is stored in a hash table that uses the identifier of the maximal family as its input, we can also affirm that each Moore family on U_{n-1} appears at least once as a maximal family.

The possibility of predicting how many times each family on U_{n-1} appears as maximal family of a family on U_n will bring us significantly close to finding a general formula, if any such formula exists.

Acknowledgment

We are grateful to Bernhard Ganter to have pointed out the misprint on the Integer Sequence Encyclopedia, and consequently confirmed our results on the number of Moore families up to isomorphism.

References

1. Habib, M., Nourine, L.: The number of moore family on n=6. Discrete Mathematics 294, 291–296 (2005)
2. Cohn, P.: Universal Algebra. Harper and Row, New York (1965)
3. Davey, B.A., Priestley, H.A.: Introduction to lattices and orders, 2nd edn. Cambridge University Press, Cambridge (1991)
4. Demetrovics, J., Libkin, L., Muchnik, I.: Functional dependencies in relational databases: A lattice point of view. Discrete Applied Mathematics 40(2), 155–185 (1992)
5. Duquenne, V.: Latticial structure in data analysis. Theoretical Computer Science 217, 407–436 (1999)
6. Ganter, B., Wille, R.: Formal concept analysis. Mathematical Foundation. Springer, Heidelberg (1999)
7. Birkhoff, G.: Lattice Theory, 3rd edn. American Mathematical Society (1967)
8. Moore, E.: Introduction to a form of general analysis. Yale University Press, New Haven (1910)
9. Caspard, N., Monjardet, B.: The lattices of closure systems, closure operators, and implicational systems on a finite set: a survey. Discrete Applied Mathematics 127, 241–269 (2003)
10. Burosh, G., Demetrovics, J., Katona, G., Kleitman, D., Sapozhenko, A.: On the number of databases and closure operations. Theoretical Computer Science 78, 377–381 (1991)
11. Higuchi, A.: Note:lattices of closure operators. Discrete Mathematics 179, 267–272 (1998)
12. Medina, R., Nourine, L.: Algorithme efficace de génération des ideaux d'un ensemble ordonné. Compte rendu de l'Académie des sciences, Paris T.319 série I, pp. 1115–1120 (1994)
13. McKay, B.D.: Isomorph-free exhaustive generation. J. Algorithms 26(2), 306–324 (1998)
14. Ganter, B.: Finding closed sets Under Symmetry. FB4-PrePrint 1307, TH Darmstadt (1990)

Appendix

Proposition 1. *Let $\mathcal{M}$ be a Moore family on U_n and $\Phi \in Sym_n$ be a permutation, then $\Phi(\mathcal{M})$ is a Moore family on U_n.*

Proof. Let us show that $\Phi(\mathcal{M})$ is a Moore family.

- $\emptyset \in \mathcal{M}$, $\Phi(\emptyset) = \emptyset$ thus $\emptyset \in \Phi(\mathcal{M})$.
- Let M_1 and $M_2 \in \mathcal{M}$ and $\Phi(M_1)$ and $\Phi(M_2) \in \Phi(\mathcal{M})$. We have $\Phi(M_1) \cup \Phi(M_2) = \Phi(M_1 \cup M_2)$ by definition of $\Phi()$. Since $\mathcal{M}$ is closed $M_1 \cup M_2 \in \Phi(\mathcal{M})$. Then $\Phi(M_1 \cup M_2) \in \Phi(\mathcal{M})$.

So $\forall \Phi(M_1)$ and $\Phi(M_2) \in \Phi(\mathcal{M})$, $\Phi(M_1) \cup \Phi(M_2) \in \Phi(\mathcal{M})$.

$\Phi(\mathcal{M})$ contains the empty set and is closed by union, $\Phi(\mathcal{M})$ is a Moore family. □

Proposition 2. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n with $n-1 \in M$ for all $M \in \mathcal{M}_{sup} \setminus \{\emptyset\}$. Then there exists a* ***unique Moore family*** *$\mathcal{M}_{max}$ on U_{n-1} compatible with $\mathcal{M}_{sup}$ such that all Moore families on U_n, whose the restriction to its elements containing $n-1$ corresponds to $\mathcal{M}_{sup}$, can be written $\mathcal{M}_{sup} \cup \mathcal{M}_{inf}$, with $\mathcal{M}_{inf} \subseteq \mathcal{M}_{max}$.*

Proof. Let $\mathcal{M}$ be a Moore family on U_n with $\mathcal{M} = \mathcal{M}_{sup} \cup \mathcal{M}_{inf}$.

A) Let us show that for any Moore family $\mathcal{M}'_{inf}$ on U_{n-1} with $\mathcal{M}'_{inf} \subseteq \mathcal{M}_{inf}$ then $\mathcal{M}_{sup} \cup \mathcal{M}'_{inf}$ is a Moore family.

- $\emptyset \in \mathcal{M}'_{inf}$, then $\emptyset \in \mathcal{M}_{sup} \cup \mathcal{M}'_{inf}$;
- Let M_1 and $M_2 \in \mathcal{M}_{sup} \cup \mathcal{M}'_{inf}$; 3 cases can occur :
  1. If M_1 and $M_2 \in \mathcal{M}_{sup}$, since $\mathcal{M}_{sup}$ is a Moore family $M_1 \cup M_2 \in \mathcal{M}_{sup}$ and then $M_1 \cup M_2 \in \mathcal{M}_{sup} \cup \mathcal{M}'_{inf}$;
  2. If M_1 and $M_2 \in \mathcal{M}'_{inf}$, since $\mathcal{M}'_{inf}$ is a Moore family $M_1 \cup M_2 \in \mathcal{M}'_{inf}$ and then $M_1 \cup M_2 \in \mathcal{M}_{sup} \cup \mathcal{M}'_{inf}$;
  3. If $M_1 \in \mathcal{M}_{sup}$ et $M_2 \in \mathcal{M}'_{inf}$. Since $\mathcal{M}'_{inf} \subseteq \mathcal{M}_{inf}$ and since for all $M \in \mathcal{M}_{inf}$, $M_1 \cup M_2 \in \mathcal{M}_{sup}$ ($M \cup M_1$ contains n) then $M_1 \cup M_2 \in \mathcal{M}_{sup}$. And so $M_1 \cup M_2 \in \mathcal{M}_{sup} \cup \mathcal{M}'_{inf}$;

B) Let us show that for all Moore families $\mathcal{M}'_{inf}$ and $\mathcal{M}''_{inf}$ on U_{n-1} such that $\mathcal{M}_{sup} \cup \mathcal{M}'_{inf}$ and $\mathcal{M}_{sup} \cup \mathcal{M}''_{inf}$ are Moore families then $C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \cup \mathcal{M}_{sup}$ is a Moore family with C an union closed operator.

- $\emptyset \in \mathcal{M}'_{inf}$, then $\emptyset \in C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \cup \mathcal{M}_{sup}$;
- Let M and $M' \in C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \cup \mathcal{M}_{sup}$; The only difficult case corresponds to the case $M \in \mathcal{M}_{sup}$ and $M' \in C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \setminus \mathcal{M}'_{inf} \setminus \mathcal{M}''_{inf}$. Since $\mathcal{M}'_{inf}$ and $\mathcal{M}''_{inf}$ are Moore families then there exists m_1 in $\mathcal{M}'_{inf}$ and m_2 in $\mathcal{M}''_{inf}$ such that $M' = M_1 \cup M_2$ and :

1. if $M_1 \in \mathcal{M}'_{inf}$ then $\forall M \in \mathcal{M}_{sup}$ we have $M \cup M_1 \in \mathcal{M}_{sup}$;
2. if $M_2 \in \mathcal{M}''_{inf}$ then $\forall M \in \mathcal{M}_{sup}$ we have $M \cup M_2 \in \mathcal{M}_{sup}$;

And since $\mathcal{M}_{sup}$ is a Moore family $M_1 \cup M_2 \cup M = M' \cup M$ belongs to $\mathcal{M}_{sup}$

We conclude that for all couple of elements M and M' in $C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \cup \mathcal{M}_{sup}$, $M \cup M'$ belongs to $\mathcal{M}_{sup}$. So $C(\mathcal{M}'_{inf} \cup \mathcal{M}''_{inf}) \cup \mathcal{M}_{sup}$ is a Moore family.

From B) we have that the set of all compatible Moore families with a given family $\mathcal{M}_{sup}$ owns only one maximal element denoted $\mathcal{M}_{max}$ such that for all families $\mathcal{M}_{inf}$ included in or equal to $\mathcal{M}_{max}$, the join of $\mathcal{M}_{sup}$ and $\mathcal{M}_{inf}$ is a Moore family. □

Proposition 3. *Let $\mathcal{M}_{sup}$ be a Moore family on U_n (all its elements containing n), the algorithm 1 computes the maximal family $\mathcal{M}_{max}$ compatible with $\mathcal{M}_{sup}$ and returns the vector which corresponds to the family $\mathcal{M}_{sup} \cup \mathcal{M}_{max}$.*

Proof. First of all let us say the the order in which element are considered follows the property : Let $M \in 2^{U_n}$, for all $M' \in 2^{U_n}$ the element $M \cup M'$ is processed before M. If $M' \subset M$ we have $M = M \cup M'$.

Let $\mathcal{M}_{max}$ be a Moore family produced by the algorithm. Let us show that $\mathcal{M}_{max}$ is a Moore family and is maximal. By construction $\mathcal{M}_{max}$ is a Moore family. Consider the following assertion: to each step of the algorithm, if a set $M \in 2^{U_n}$ is not added to $\mathcal{M}_{max}$ then $\mathcal{M}_{sup} \cup \mathcal{M}_{max} \cup M$ is not a Moore family. Let us demonstrate the assertion reductio ad absurdum : Suppose that there exists a set $M \notin \mathcal{M}_{max}$ such that $\mathcal{M}_{sup} \cup \mathcal{M}_{max} \cup M$ is a Moore family. Then there exists a set M' processed before M, and a set M'' processed after M with $M'' = M \cup M'$. Which is in contradiction with the process' computational order. Absurd. □

Proposition 4. *Let $\mathtt{M}_n$ and $\mathtt{M}_{n+1}$ be the sets of all Moore family respectively on U_n and U_{n+1} then we have $\mid \mathtt{M}_{n+1} \mid \leq 2 * \mid \mathtt{M}_n \mid^2$.*

Proof. Each family $\mathcal{M} \in \mathtt{M}_n$ is going to produce two superior families $\mathcal{M}_{sup_1} = \{M \cup n \mid M \in \mathcal{M}\} \cup \emptyset$ and $\mathcal{M}_{sup_2} = \{M \cup n \mid m \in \mathcal{M}\} \cup \emptyset \setminus n$. Therfore, the set $\mathtt{M}_{sup_{n+1}}$ of all families $\mathcal{M}_{sup}$ contains $2* \mid \mathtt{M}_n \mid$ elements. In the worst case the maximal family associated to each family $\mathcal{M}_{sup} \in \mathtt{M}_{sup_{n+1}}$ is the Moore family 2^{U_n} of weight $\mid \mathtt{M}_n \mid$. So $\mid \mathtt{M}_{n+1} \mid \leq 2 * \mid \mathtt{M}_n \mid^2$. □

Lattice Drawings and Morphisms

Vincent Duquenne

Equipe Combinatoire et Optimisation,
CNRS & Université Pierre et Marie Curie,
175 rue du Chevaleret, 75013 Paris, France
duquenne@math.jussieu.fr

Abstract. Let $L \rightarrow H$ be a lattice homomorphism and let a "readable" drawing of H be given. It is natural to make use of it to try getting a clear(er) drawing of L. Hence, the following question is explored: How the knowledge of the congruence lattice Con(L) of L can help in getting "better" drawings of L? This will be done by proposing rank shelling procedures of $(M(Con(L),\leq)$ and will be illustrated with examples coming either from math. or social sciences.

Keywords: Lattice drawing, lattice homomorphism, congruence lattice.

1 Introduction

Since the very beginning of FCA ("Formal Concept Analysis" see [20], [16]) and more generally of the use of *lattices* (and *orders*) in Data Analysis (see [6], [10] and the applications quoted therein), the question of getting "good" / "readable" lattice drawings has been crucial. Whenever one sustains the claim that lattices are instrumental to decipher structures (*associations*, *implications*) extracted from data, better get understandable structures. Otherwise their promotion will be … hazardous.

A main evident difficulty, the lattice size can explode exponentially (regarding its input). A second one comes from the arbitrariness of "observed" lattices, so that the methods which are adapted for highly structured lattices encountered in math. cannot be of help. Here, we are dealing with arbitrary finite lattices. Besides this, defining precisely what could be a "good" drawing is not that easy, and will never receive a unique answer. It depends on the data nature and size, on its degree of structuring, and these questions are certainly context sensitive (depending on scientific contexts etc).

Now, this need for lattice drawings has been addressed specifically and this in two directions. Recently in the way of *Automated Lattice Drawing* (see [14]) by using *attraction / repulsion forces*, but all along the development of FCA, by using much more structural approaches based on several *lattice decompositions* (see [23], some of them already present in [20]). This structural attitude has been followed in developing our program GLAD (see [9]) in which *nested line diagrams* ([16] p. 75) and *gluing decompositions* ([22], [16] p. 195) have been implemented since the mid-eighties. This is also the direction followed here, with the proposal of new iterated *shelling decompositions* which are dependent on the global structure of the *congruence lattice*.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 88–103, 2010.

2 Congruence Emergency Toolkit

From classical sources (for Lattice Theory see [2], [17] and FCA see [16]), we will shortly extract a minimal set of notions and properties that are required in the sequel.

Homomorphisms are "structure preserving maps between algebraic structures". Hence for lattices, they should preserve the two operations and the order relation.

For two lattices $(L_1,\leq,\vee,\wedge)$ and $(L_2,\leq,\vee,\wedge)$, $\varphi:L_1 \to L_2$ is a (*lattice*) *homorphism* iff $\phi(x\wedge y)=\phi(x)\wedge\phi(y)$ and $\phi(x\vee y)=\phi(x)\vee\phi(y)$ hold for all $x,y\in L_1$.

Through $x\leq y \Leftrightarrow x=x\wedge y \Rightarrow \phi(x)=\phi(x)\wedge\phi(y) \Rightarrow \phi(x)\leq\phi(y)$, ϕ must actually be *isotone*.

A lattice homomorphism $\varphi:L_1 \to L_2$ quotients L_1 by a *congruence* Φ defined by $x\Phi y$ iff $\phi(x)=\phi(y)$ for $x,y\in L_1$. Φ is an equivalence on L_1 that respects the *substitution properties*: $x\Phi y$ and $u\Phi v$ implies $x\wedge u\Phi y\wedge v$ and $x\vee u\Phi y\vee v$, for $x,y,u,v\in L_1$. Now let $x\Phi y$, $x\Phi x$ and $x\Phi y \Rightarrow x=x\wedge x\Phi x\wedge y$, and similarly $x\Phi x\vee y$ holds, so that $x\Phi y$ implies $x\wedge y\Phi x\vee y$, and the Φ-classes are *convex intervals* of L_1. x/Φ denotes the class of x.

Surjective homomorphisms and congruence relations express the two sides of a single phenomenon -depending on which side of the arrow the focus is- which is made precise through the so-called *homomorphism theorems* (see [2] and [17] p. 16): for a surjective homomorphism $\varphi:L_1 \to L_2$, the *quotient lattice* $L/\Phi:=(x/\Phi \mid x\in L)$ is isomorphic to L_2 ; of particular importance here as it gives to drawings a "coherence of inheritance": for a congruence Θ on a lattice L, the congruence lattice of $L/\Theta:=(x/\Theta \mid x\in L)$ is isomorphic with the interval $[\Theta,1]$ of the congruence lattice of L.

Let Con(L) be the congruence lattice of a lattice L, ordered by set inclusion (as subsets of $L\times L$). Con(L) is a 01-sublattice of the partition lattice Part(L). Moreover (see [15]) Con(L) is *distributive* (quite a strong property among other algebras).

When Con(L)={0,1} L is called *simple*. When Con(L) has a single atom, its zero element (identity) cannot be expressed as the meet of others, in which case L is called *subdirectly irreducible*. Otherwise, when there are several atoms $\Theta_j(j\in J)$ in Con(L), thanks to distributivity, the identity can be expressed as its *minimal expression in meet-irreducible elements* (meet of all the meet-irreducible congruences that are *perspective* to the $\Theta_j(j\in J)$ in Con(L)). In this case, L is called *subdirectly reducible* and is a *subdirect product* of the irreducible *factors* $L/\Theta_j(j\in J)$, as it can be identified as a *sublattice* of their product which projects surjectively onto the factors $L/\Theta_j(j\in J)$.

Some notations will be needed for dealing concretely with congruences. Let L be a lattice, J:=J(L) and M:=M(L) its sets of *join-* and *meet-irreducible* elements, and $(J,M,\leq)$ be its *standard context* (table). For $j\in J$ and $m\in M$, $j\uparrow m$ means that m is maximal not above j, dually $m\downarrow j$ that j is minimal not below m, in which case they are *weakly perspective*. A pair of elements in $J\cup M$ is *weakly projective* if there exists a path of alternating weak perspectivities between them (ex: $j_1\uparrow m_1\downarrow j_2\uparrow m_2\downarrow j_3\uparrow m_3$...).

How to calculate Con(L)? (see [21] p. 280, [14] and http://www.latdraw.org/ for a program). For $m\in M$ let $m/\approx$ be its *weakly projective closure* in $J\cup M$. $(M(Con(L)),\leq)$ is isomorphic to the ordered set (by dual inclusion) of all weakly projective closure of $m\in M$ (in words, this is just the consequence of "heredity of not collapsing").

Finally, congruences on a lattice L are represented into the Hasse diagram of L (representing its *cover relation* obtained by *transitive reduction* of $(L,\leq)$, while the

latter is reconstructed by *transitive closure*). Lattice homomorphism corresponds to homomorphism of the covering relation, hence their importance here for diagrams.

3 On Frattini Congruences

As a motivating example let consider the *permutohedron* Perm(4), that is the set of all *permutations* on {1,2,3,4} rooted at 1234 (see Fig. 1), and ordered by *transposing* adjacent elements (other orders are sometimes considered). The structure of Perm(n) was extensively studied (see [12], with subsequent papers in literature on *weak orders* and *multinomial lattices*) up to characterizing iteratively its *standard context* (J(Perm(n)),M(Perm(n)),≤) out of (J(Perm(n-1)),M(Perm(n-1)),≤) through a simple copy-and-paste procedure, which could also iteratively characterize the *arrow-graph* expressing the *weak-perspectivities* between its join- and meet-irreducible elements.

This permitted the construction of the congruence lattice Con(Perm(4)) (see Fig. 2) through a characterization of its ordered set of meet-irreducible elements / congruences (M(Con(Perm(4))),≤), thanks to distributivity. From its structure it could be derived that it has 2**(n-2)=4 minimal elements (see [12], p. 80) so that Perm(4) is a subdirect product with as many hence four irreducible factors. In Fig. 2, these minimal meet-irreducible congruences are labelled by "I","K","M", and "O". Consequently, Perm(4) has four subdirectly irreducible factors that are defined by the *order filter* that they generate in (M(Con(Perm(4))),≤), namely:

IEJCFL, MENCFL, KGJCFL and OGNCFL, respectively.

CLAIM 1. The notion of subdirect product is clear and neat (but requires training since one must characterize which is the sublattice of the product see [21]). The uniqueness of … "subdirect product of irreducible factors" provides some reassuring canonicity. But despite this, in such situations as above where the factors of Perm(4) share a lot of attributes ("C","F","L" globally and more pairwise), they will not help in obtaining "understandable" drawings of Perm(4), since the product of factors becomes so big compared with Perm(4) (4×6=24 / 11 meet-irreducibles in Perm(4)).

Here we will call *Frattini congruence* of a lattice the intersection of its *coatomistic* (lower covers of the unit) congruences, just like a *Frattini sublattice* (see [18]) of a lattice is the intersection of its maximal proper sublattices (beware that in literature another definition involving congruences on sublattices was also considered).

The interest of our Frattini congruences will be better understood if we first examine under which conditions on L its congruence lattice Con(L) is Boolean, that is –being already distributive- when its Frattini congruence equals identity in Part(L). This is the case: iff the weak-projectivity relation on J(L)∪M(L) is symmetric, iff the arrow-graph of weak-perspectivities on J(L)∪M(L) is the disjoint union of strongly connected components, iff L is a subdirect product of simple factors. These lattices are called *weakly-modular* and comprise *modular*, hence *distributive*, and *relatively complemented* lattices which do have Boolean congruence lattices.

These above conditions express that for such a lattice L, M(L) and J(L) are mutually splitted into blocks of pairwise symmetrically weakly-projective elements, and that otherwise any pair of elements belonging to two distinct blocks are not weakly-projective (nor –perspective). Here is a harsh contrast, between what we will call *full symmetric wp-dependence* (within the blocks) and *full wp-independence*.

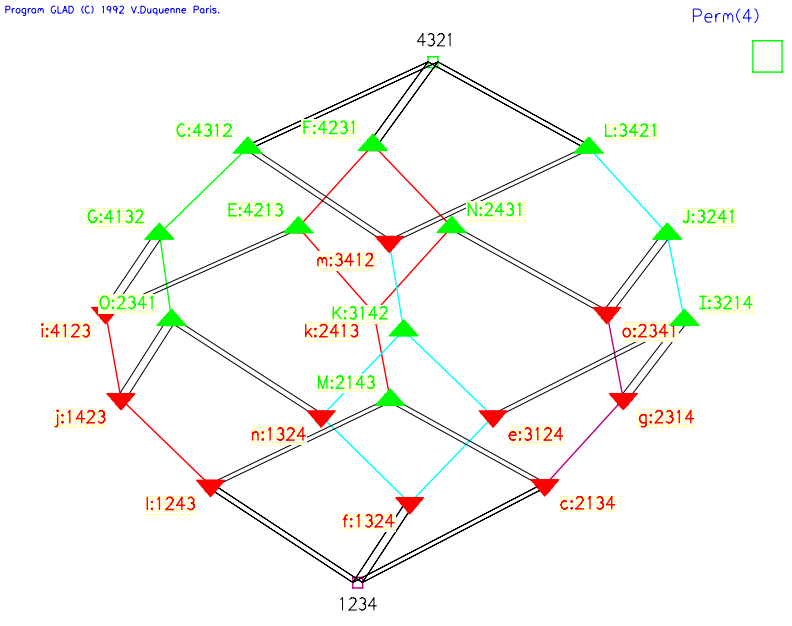


Fig. 1. Decomposition of the permutohedron Perm(4) by its Frattini / Glivenko congruence (congruence classes are colored with single lines, while they are connected with double lines)

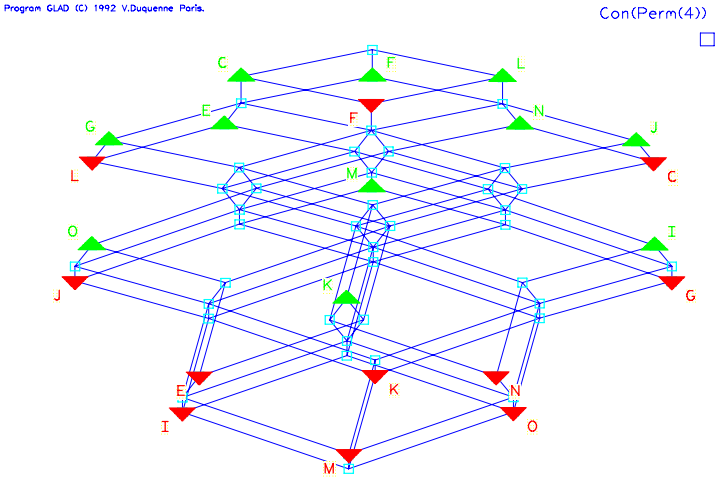


Fig. 2. The congruence lattice of the permutohedron Perm(4) was initially (see [12]) calculated and derived through characterizing of the set of its meet-irreducible elements: (M(Perm(4)),≤)

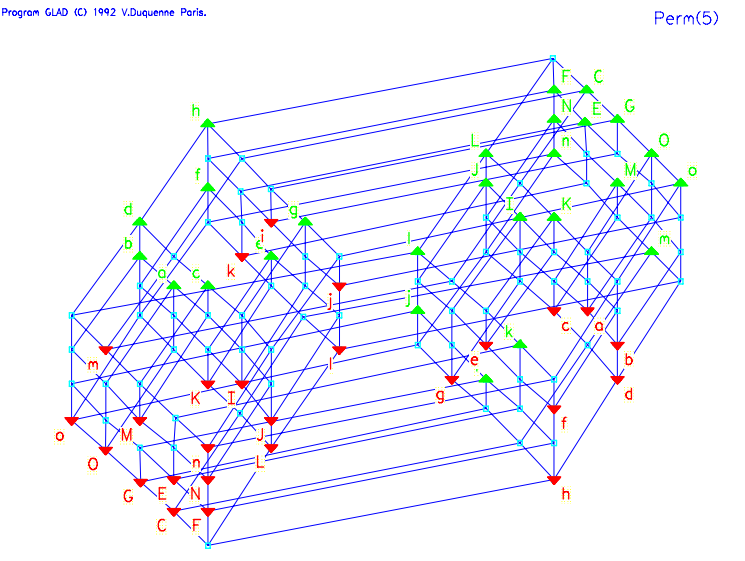


Fig. 3. The permutohedron Perm(5) (order-) embedded into a product of chains: 5x4x3x2

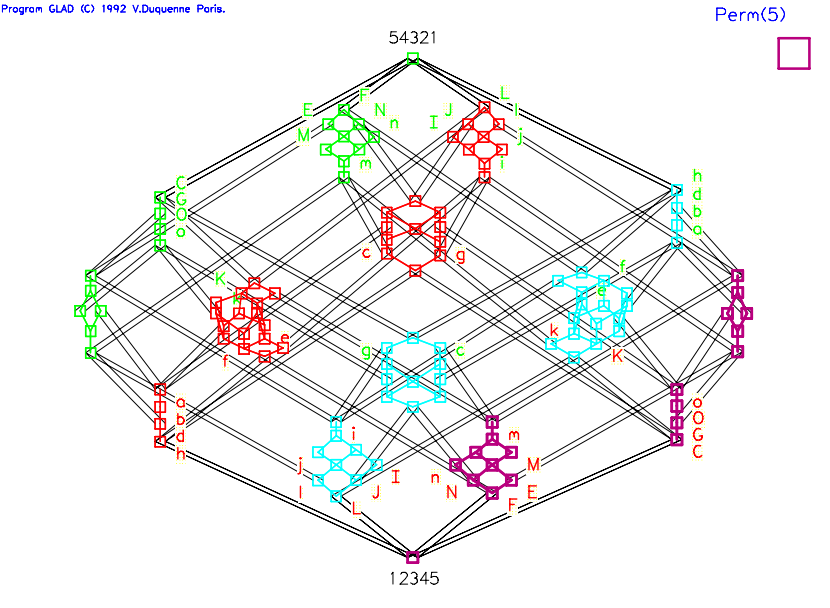


Fig. 4. The permutohedron Perm(5) quotiented by its Frattini / Glivenko congruence

CLAIM 2. Considering the Frattini congruence Φ_L of an arbitrary L will similarly identify those meet / join-irreducible elements –think of objects / attributes in data analysis- that are fully wp-dependent / independent. Moreover Considering L/Φ_L , and drawing L according to an "understandable" drawing of L/Φ_L would clarify L.

Coming back to our motivating example Perm(4), let denote by Φ its Frattini congruence (also known as *Glivenko congruence* for such *complemented* lattice and interpreted as "sharing the same complement"). Φ(Perm(4)) is a Boolean lattice 2**3. Fig. 1 provides quite a clarified drawing of Perm(4)/Φ along this Boolean lattice (compare with [14] p. 127 which is very similar in shape although it was obtained through a completely different à la *spring embedder* method). Similarly, Fig. 3 provides an (order) embedding of Perm(5) into a product of chains (compare with [16] p. 56), while Fig. 4 gives a more symmetric view of Perm(5) through the decomposition by its Frattini congruence into a Boolean 2**4 quotient lattice.

It may seem reasonable that this method can clarify lattices coming from math, just by respecting the symmetries like the left-right symmetry of the rooted Perm(n). But dealing with arbitrary lattices coming out of FCA & data analysis, a real question is: Will such decompositions by Frattini congruences clarify lattices coming from data?

A first illustration coming from social networks is given in Fig. 5. The original study (see [19]) focus on global structures of overlapping relations among Brazilian youth organizations and their projects, during the so-called "1992 impeachment movement". The data is a small 29×8 *context* (binary relation / matrix). Let us call for short L its concept (/ Galois) lattice. Its congruence lattice Con(L) (see Fig. 6) shows that L is subdirectly irreducible, and that its unique atom is equal to its Frattini congruence Φ_L. The quotient lattice L/Φ_L is isomorphic to a 2×3 product of chains, while interestingly the five remaining attributes that are collapsed by Φ_L belong to the top class: the complex structure of wp-dependency among these attributes is simply unfolded downwards by the three wp-independent attributes "C","F","G" into a quite simple distributive product of chains. Of course, it would be interesting that the specialists could interpret these facts regarding their knowledge on associations.

A second application comes from the pedagogy of mathematics (see [4]) and is represented in Fig. 8. The data was previously analyzed through *implications* and *gluing decompositions* (see [11]). The congruence lattice (see Fig. 7) is just a bit more complex, with the grafting a three element chain below the Frattini congruence. Here again as before L/Φ_L is a distributive lattice generating a clear split between wp-independent / dependent attributes and simplifying drastically the drawing.

A third example comes from a study on partial Down syndrome in genetics (see [5] and [13]), where the data describes young patients in terms of triplication of genetic bands in Chromosome 21. The quotient lattice L/Φ_L is again distributive, and interestingly the attributes which are collapsed by Φ_L belong to the top class that corresponds to the center of Chromosome 21, which is unfolded at left / right, with a bifurcation for the latter, generating the three dimensional quotient lattice.

Hence, Frattini congruences Φ_L help in clarifying these three small examples, but they share that Φ_L are not far away from the identity in their congruence lattices, and all quotient lattices are distributive. What can be done with more complex examples?

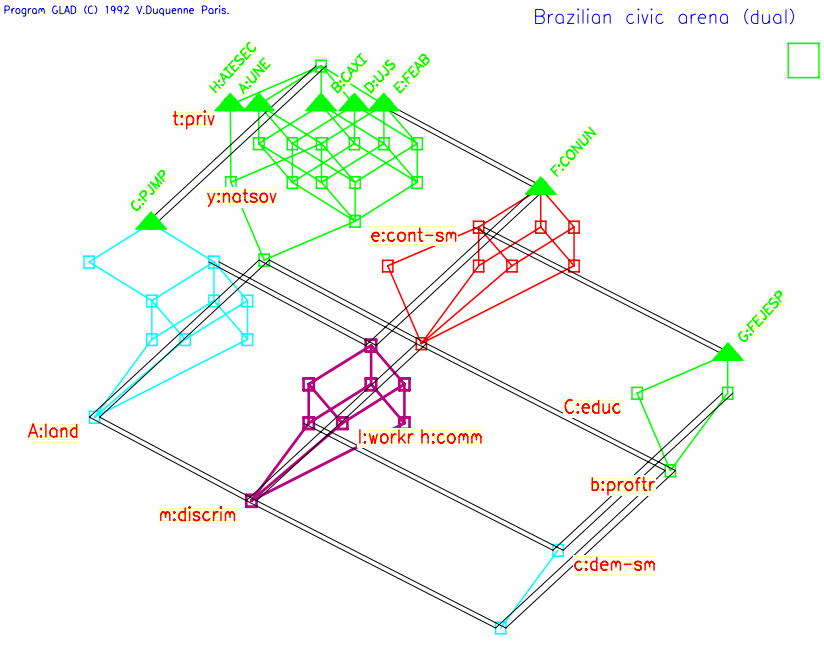


Fig. 5. A lattice coming from a study (see [19]) on Brazilian youth (organizations × projects). Its Frattini congruence quotients the lattice onto a simple distributive 2×3 product of chains, by collapsing together five pairwise weakly-projective attributes within the top class

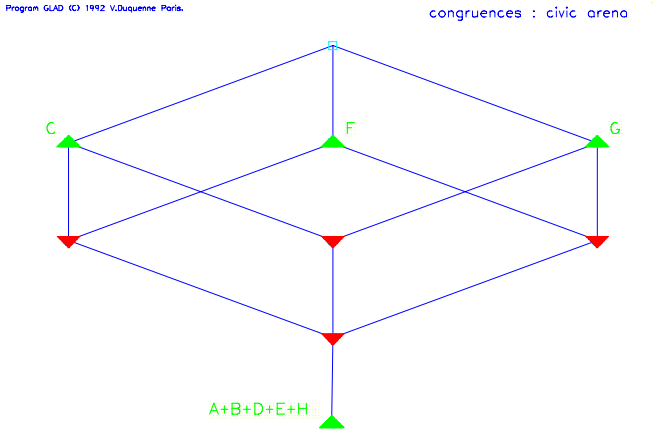


Fig. 6. Its congruence lattice splits three "independent" / five weakly-projective attributes

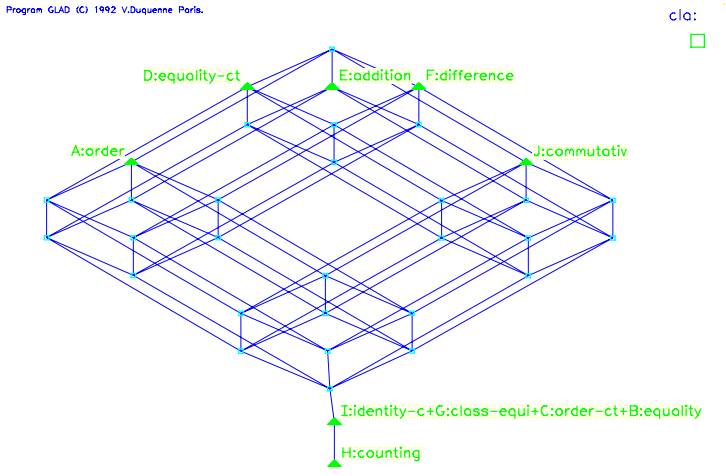


Fig. 7. A congruence lattice from a study ([4,11]) on math. teaching. The data describes whether children master properties of natural numbers and operations (addition, counting…). The aim is to evaluate implications between them, and locate the children in the lattice.

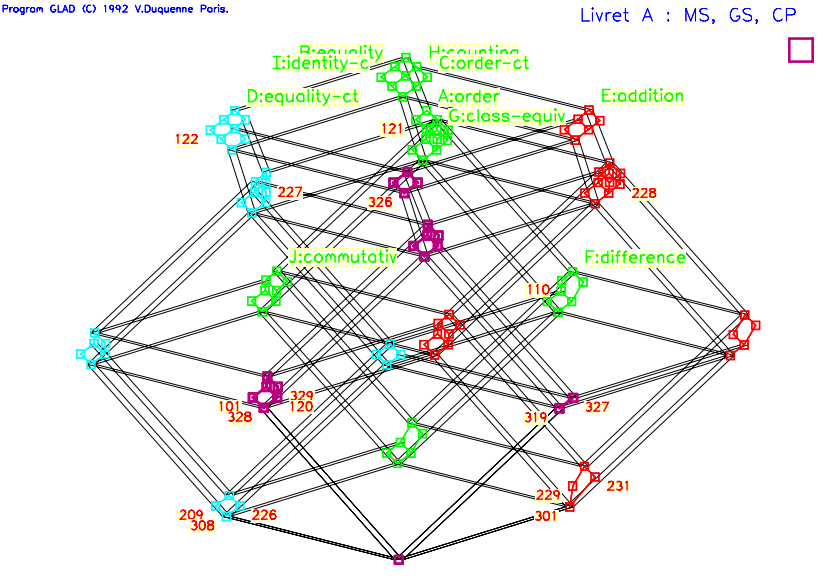


Fig. 8. The Frattini congruence generates a similar split between attributes and quotients into a distributive lattice that therefore reveals a simple hierarchy between “independent” attributes

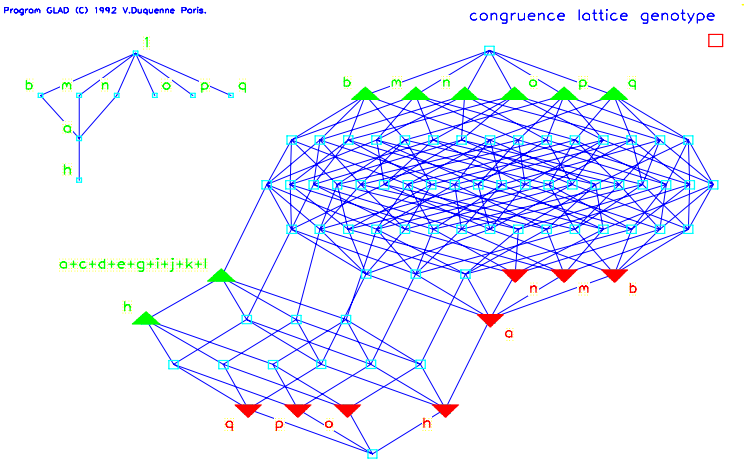


Fig. 9. A congruence lattice coming from genetics, with its meet irreducible elements. The data describes patients in terms of triplication of genetic bands in Chromosome 21 (see [5,13]).

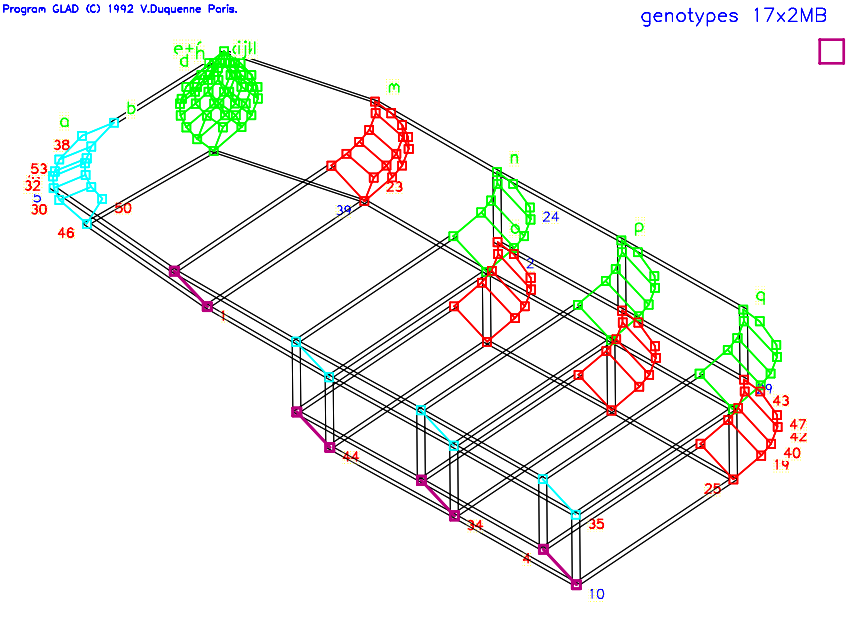


Fig. 10. Its Frattini congruence collapses the left "a" & center "cd…kl" of chromosone 21 within the top class while remaining bands unfold & project the lattice on a distributive factor

4 Nested Congruences and Diagrams

Our motivating example will this time come from math. and even closer from "FCA on math". In his paper on "subdirect product decomposition of concept lattices" (see [21]) R. Wille gave a nice drawing from an example describing homomorphisms of partial algebras by abstract properties. It was imitated in Fig. 11. It could be said that the drawing of this lattice –say L- is clear, constructed under the control of (M(L),≤) by applying the parallel law of *additive line diagrams* (see [16] p. 76).

This lattice was certainly chosen because it is a subdirect product of four irreducible factors which are not too complex, and is reconstructed by a kind of fusion (see [21] p. 229) of their *scaffoldings* which are some kind of distinguished generating subsets (for the notion of *partial sub-semilattice*). We bettered this by showing that it is also a same kind of fusion of the (*minimal* for this kind of construction through partial subsemilattices) factor *meet-cores* (see [7] , [8] Th. 4 p. 137).

CLAIM 3. If these (re)constructions of a subdirect product L are clear on an abstract level and can be programmed, they don't help for clarifying the drawing of L because the product of the factors is generally big, and these fusion procedures are complex (see [7] p. 229-233) since carrying all the projections of L onto the factors.

To try going a bit further, Con(L) and (M(Con(L)),≤) have been represented in Fig. 12. (M(Con(L)),≤) has four minimal elements, as expected, ("j","g","h","f"). The Frattini congruence Φ_L generates a quotient lattice L/Φ_L (see Fig. 13) which is Boolean and meet-generated along maximal elements of M(Con(L)),≤): "a","b","d". Now, seven attributes are left, so that in such situation where Con(L) is far away from being Boolean, we propose to reiterate the process along a "shelling procedure".

For M:=M(L), let (M, $\geqq$) be the preorder induced on M by the *weakly projective closure*: $m_1 \geqq m_2 \Leftrightarrow m_1/\approx \supseteq m_2/\approx$. The equivalence classes of this preorder are ordered isomorphically to (M(Con(L)),≥). Let us call in short $M_R:=\{M_r / r=1,2,\ldots\}$ *shelling partition* of M that is obtained by a "*rank down shelling*" of (M(Con(L)),≥):

$M_1=\max(M, \geqq)$, $M_2=\max(M \setminus M_1, \geqq),\ldots,M_r=\max(M \setminus \cup\{M_s/s=1,r-1\}, \geqq)$.

CLAIM 4. The "*rank down shelling*" of (M(Con(L)),≥) procedure leads to define:
- *A canonical representation of L*, by drawings which refine a drawing of the Frattini quotient L/Φ_L by refining the classes iteratively along the shelling partition M_R.
- *A canonical presentation of the "standard context"* (J(L),M(L),≤) of L (see Fig. 15) with the *arrow graph* of weak perspectivities, which is quotiented into strongly connected components by the map $M(L) \to M(Con(L))$. The elements $m \in M$ are ordered from left to right along the shelling partition M_R, and J(L) and M(L) are permuted so that the strongly connected components be connected into the context.
The claim is that these (re)presentations put what is clear (*full wp-independence / dependence*…) in front line, and nest the remaining complexity in a canonical order.

Such a "canonical" drawing is represented in Fig. 13. To get an even more readable drawing, we erased in Fig. 14 the nested boxes –as is often done for "nested line diagrams"-, and distinguish / reconstruct the three shelling levels by using three different distances between attributes $m \in M$ and their unique upper covers m* in L.

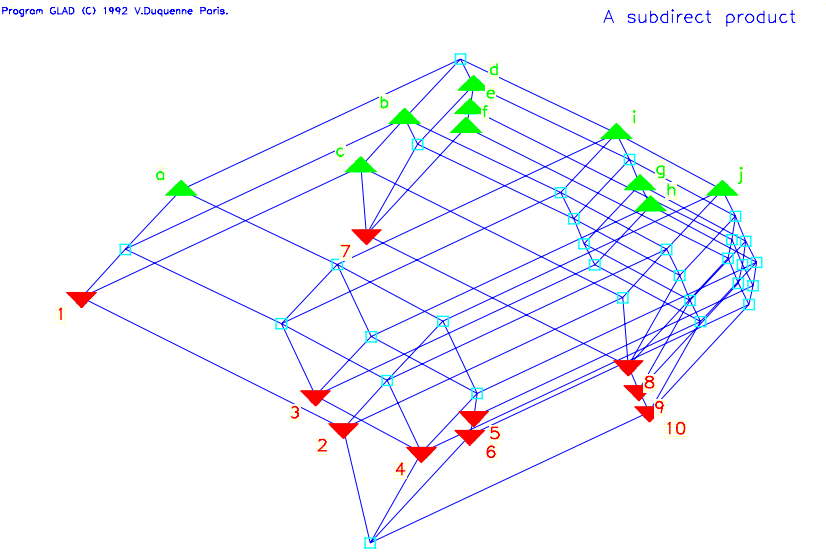


Fig. 11. A subdirectly reducible lattice L coming from mathematics (description of homomorphisms on partial algebra see [21], from which the lattice drawing has been borrowed)

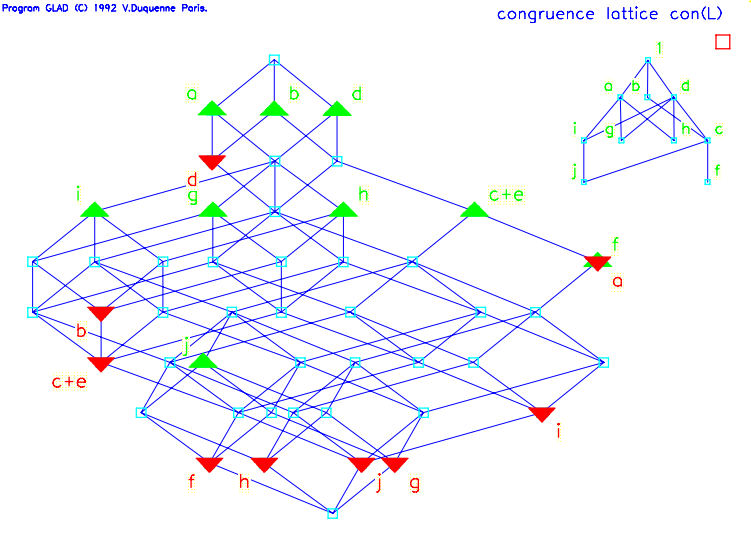


Fig. 12. The order (M(Con(L)),≤) has four minimal elements so that L is the subdirect product of four irreducible factors (capturing respectively all attributes above “j”, “g”, “h” and “f”)

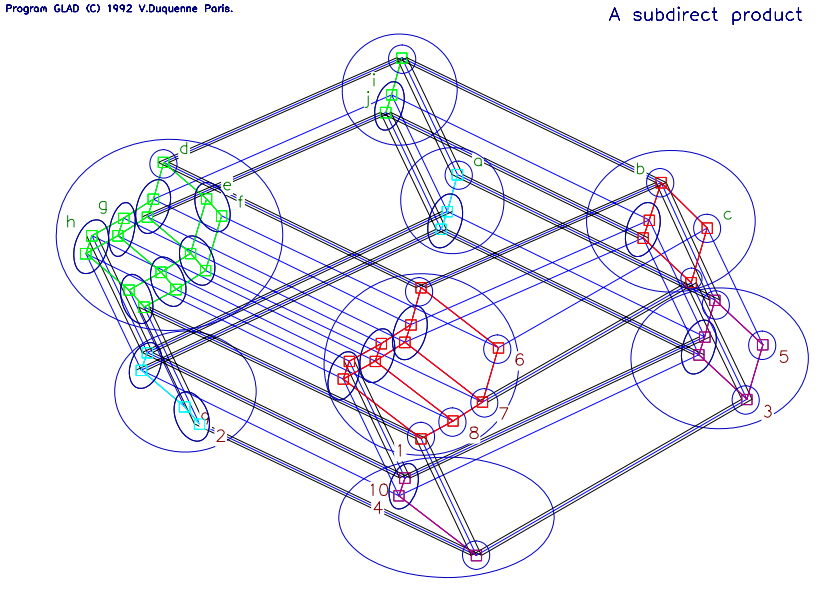


Fig. 13. A canonical presentation of L by "*rank down shelling*" of (M(Con(L)),≥): the Frattini congruence Φ_L generates a quotient lattice L/Φ_L which is Boolean and meet-generated along: "a","b","d" (first level of boxes). The L/Φ_L-classes are refined the same way at the next level.

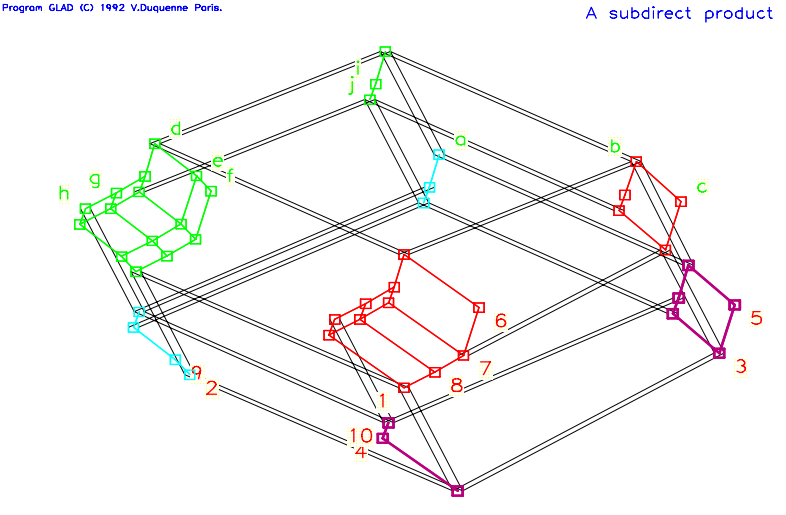


Fig. 14. The same canonical presentation of L by "*rank down shelling*" of (M(Con(L)),≥), where the drawing is simplified by removing boxes and lines as often done for nested line diagrams. The three shelling levels can be easily identified by using three different distances.

↕	x	x	x	x	x	x	x	x	X		1
x	↕	x	↓	x	x	x	X	x	X	J_1	2
x	x	↕	x	↓	x	↓	↓	↓	X		3
x	x	x	↕	↕	x	x	X	↓	X		4
x	x	↑	x		↕				↓		5
↑	x	x	x	x	↕			x	↓	J_2	6
↑	x	x	x	x	x	↕		x	X		7
↑	x	x	x	x	x	x	↕	x	X		8
x	↑	x		x	x	x	x	↕	X		9
x	x	x	↑	↑	x	x	x		↕	J_3	10
	M_1				M_2				M_3		
a	b	d	c	e	i	g	h	f	j		

Fig. 15. A canonical presentation of the "standard context" (J(L),M(L),≤) with the arrow graph along the *rank down shelling* of (M(Con(L)),≥), which defines the partition (M_1,M_2,M_3). J(L) and M(L) have been permuted so that the graph strongly connected components be connected. There are obviously no up-arrows (down-arrows) above (left of) the subtables J_rxM_r (r=1,2,3).

5 On the "Importance of Morphisms"

This question has been advocated for our present and pragmatic concerns regarding lattice drawing clarification, but it was raised initially at the general level of philosophy of mathematics. There, many authors (see [1] p. 209) have agreed that:

"From Dedekind, through Noether, and the work of Eilenberg and Mac Lane, the fact has clearly emerged that mathematical structure is determined by a system of objects *and their mappings*, rather than by any specific features of mathematical objects viewed in isolation. To a great degree, the structural approach in modern mathematics is characterized by increased attention to (system of) *mappings*, and the idea that mathematical objects are determined by their *'admissible' transformations*."

Moreover, as attested by G. Birkhoff (see [3, p. 773], among S. Mac Lane, N. Bourbaki etc), in half of a century, this shift developed widely with enthusiasm to reformulate algebra and even to call for rethinking the foundations of mathematics:

"The tidal wave generated by enthusiasm about abstract algebra had wider repercussions. (…) The "universal" approach to algebra, which I had initiated in the 1930's and 1940's stressing the role of lattices, was developed much further in two important books by Cohn and Grätzer. In a parallel development, Lawvere (1965) proposed "The category of categories as a foundation for mathematics," beginning with the statement: *'In the mathematical development of recent decades one sees clearly the rise of the conviction that the relevant properties of mathematical objects are those which can be stated in terms of their abstract structure rather than in terms of the elements which the objects were thought to be made of. The question thus naturally arises whether one can give a foundation for mathematics (…) in which classes and membership in classes do not play any role.*' "

Now, this abstract movement could have been wild and dominating in the 70's. At the end of a lecture, I remember a great professor asking "why don't you embed this into Category Theory?", which has had terrified me for twenty years, up to relaxing only after seeing it as leading in a "list of best stupid questions in Mathematics".

Here, more modestly, it has been advocated and illustrated that *congruence relations* and *lattices* -although abstract- can be instrumental in taming the complexity of lattice drawings. This has been done through a series of claims and examples.

A first motivating example has been the permutohedron Perm(n), on which the *Frattini congruence* defines a Boolean lattice of classes, each of which can be interpreted as putting together lattice elements *sharing the same complements*. This decomposition gives new views / perspectives of Perm(n). To get such interpretable congruence that gives new insights of a lattice should be most useful for data analysis.

Hence, three examples from social sciences and genetics demonstrate that the Frattini congruence can help in extracting –hopefully big- distributive factors that are helpful in simplifying lattice drawings thanks to the nice properties of distributivity.

For more complex examples for which congruence lattices depart from being Boolean, we introduced a procedure of *rank down shelling* of (M(Con(L)),≥), that decomposes congruence classes iteratively along a chain of nested congruences. The kind of drawing clarification that can be obtained is illustrated on an example coming from math. The general idea behind this procedure is to put what is clear, the *contrast* between full weakly projective *dependence* / in*dependence* in front line, and to nest the remaining complexity in a *linear order* of such embedded contrasts, which is determined by the *congruence lattice global structure*: all the *morphisms* are collectively used to shape an appropriate … *morphology* to the lattice. This hierarchy of contrasts is very close to the "hierarchy of contradictions" often met in philosophy: this follows a balanced claim for more "concreteness & abstraction, nothing less!".

6 Added in Proofs

Congruences can be used for formalizing dualities like *identification* / *distinction* via the *collapsing* / *splitting* of lattice elements -under the substitution properties and both operation respects. Hence, the above *rank down shelling* procedure gives *priority to distinction* (splitting). But there are situations where identification (collapsing) should prevail…This leads to consider a dual *up shelling* procedure scanning through the order (M(Con(L)),≥) –as well as the induced preorder (M, $\underline{\geq}$)- from bottom to top.

The resulting drawing is represented in Fig. 15, based on the up shelling partition of M(L): M_1=ghfj, M_2=cei, M_3=abd. As compared with Fig. 13, the "boxes" should now be more easily read from the smaller (inside) to the bigger (outside). If the two drawings are not that different (in this particular case), here the priority is to collapse pairs "having no other consequences" for collapsing, by considering them as similar. The shelling procedure provides a canonical ordering of such similarities. Comparing the two logic of *distinction* / *identification* should now be done on appropriate data...

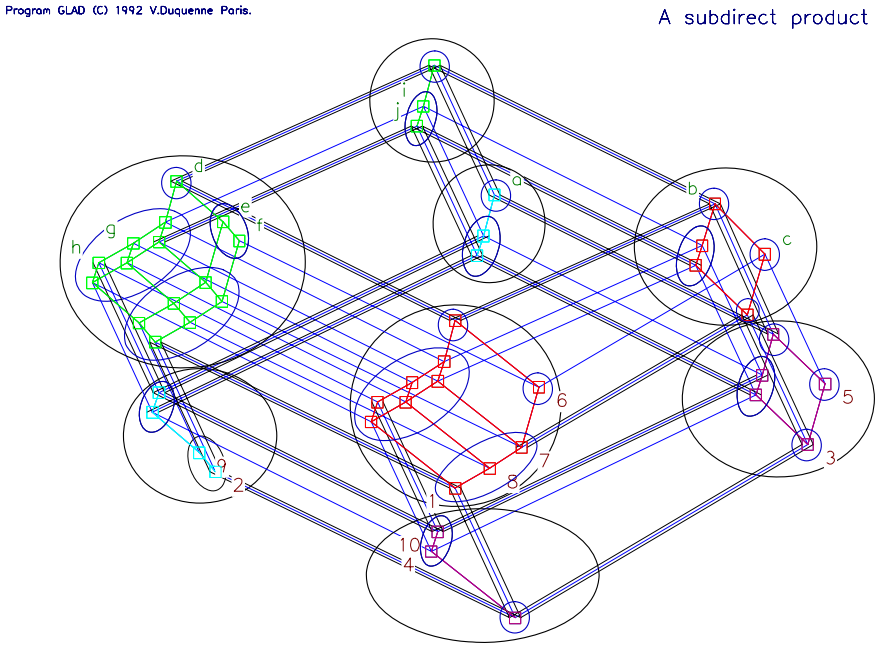


Fig. 15. Another canonical presentation of L by "*rank up shelling*" of (M(Con(L)),≥). Here the *shelling partition* is: M_1=min(M, ≧), M_2=min(M \ M_1, ≧),…,M_r=min(M \ ∪{M_s/s=1,r-1}, ≧).

Acknowledgments

A preliminary draft of this work on lattice drawings was presented at Institut Henri Poincaré, Paris, 04/2009, for the meeting: "*Pretty Structure, Existential Polytime and Polyhedral Combinatorics (Celebrating Jack Edmonds 75th birthday)*", http://www.ecp6.jussieu.fr/images/celabrating%20Jack%20Edmonds%20Birth.pdf .
Happy birthday Jack! Thanks to you and the other Waterloo survivors –Henry Crapo and Adrian Bondy- reunified these days in Paris around so interesting discussions.

Thanks are also due to the referees for their remarks and comments. In particular a proposal has been to drop out our assumption of homomorphism *surjectivity*, with the following motivation: for an homomorphism ϕ:L → H and a "nice" drawing of H, it is possible that removing the elements of H \ ϕ(L) makes the drawing becoming "ugly". This extension would give more flexibility and should be explored … in the future.

References

[1] Awodey, S.: Structure in Mathematics and Logic: a categorical perspective. Philosophia Mathematica 4, 209–237 (1996)
[2] Birkhoff, G.: Lattice Theory, 3rd edn. American Mathematical Society Colloq. Publ, vol. 25. Amer. Math. Soc., New York (1967) (1st edn. 1940)

[3] Birkhoff, G.: Current Trends in Algebra. The American Mathematical Monthly 80, 760–782 (1973)
[4] Charron, C.: Ruptures et continuités dans la construction des nombres, Phd Thesis, Université Paris V-René Descartes, Paris (1998)
[5] Delabar, J.M., Theophile, D., Rahamani, Z., Chettouh, Z., Blouin, J.L., Prieur, M., Noel, B., Sinet, P.M.: Molecular mapping of twenty-four features of Down Syndrome on chromosome 21. European Journal of Human Genetics 1, 114–124 (1993)
[6] Duquenne, V.: What can lattices do for experimental designs? Mathematical Social Sciences 11, 243–281 (1986)
[7] Duquenne, V.: Contextual implications between attributes and some representation properties for finite lattices. In: Ganter, B., Wille, R., Wolff, K.E. (eds.) Beitrage zur Begriffsanalyse, Mannheim, pp. 213–239 (1987)
[8] Duquenne, V.: On the core of finite lattices. Discrete Mathematics 88, 133–147 (1991)
[9] Duquenne, V.: GLAD (General Lattice Analysis & Design): a FORTRAN program for a glad user, ORDAL 96 (Rival, I. (ed.), Ottawa, http://www.csi.uottawa.ca
[10] Duquenne, V.: Latticial structures in Data Analysis, ORDAL 96: Order and decision-making (Rival, I. (ed.), Ottawa and Theoretical Computer Science 217, 407–436 (1999), http://www.csi.uottawa.ca
[11] Duquenne, V.: What can lattices do for teaching math.? In: Diatta, J., Eklund, P., Liquière, M. (eds.) CLA 2007, pp. 72–87 (2007) posted at CEUR-WS proceedings 331, http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-331/
[12] Duquenne, V., Cherfouh, A.: On permutation lattices. Mathematical Social Sciences 27, 73–89 (1993)
[13] Duquenne, V., Chabert, C., Cherfouh, A., Delabar, J.-M., Doyen, A.-L., Pickering, D.: Structuration of phenotypes / genotypes through Galois lattices and implications. In: Mephu Nguifo, E., et al. (eds.) Proc. of ICCS 2001-CLKDD 2001, Stanford 07/2001. Applied Artificial Intelligence, vol. 17, 21-32, pp. 243–256 (2003)
[14] Freese, R.: Automated lattice drawing. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 112–127. Springer, Heidelberg (2004), http://www.latdraw.org/
[15] Funayama, N., Nakayama, T.: On the distributivity of a lattice of lattice-congruences. Proc. Imp. Acad. Tokyo 18, 553–554 (1942)
[16] Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin (1999)
[17] Grätzer, G.: The congruences of a finite lattice: a proof-by-picture approach. Birkhäuser, Basel (2006)
[18] Koh, K.-M.: On the Frattini sublattice of a lattice. Algebra Universalis 1, 104–116 (1971)
[19] Mische, A., Pattison, P.: Composing a civic arena: Publics, projects, and social settings. Poetics 27, 163–194 (2000)
[20] Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered sets, pp. 445–470. Reidel, Dordrecht (1982)
[21] Wille, R.: Subdirect decomposition of concept lattices. Algebra Universalis 17, 275–287 (1983)
[22] Wille, R.: Complete tolerance relations of concept lattices. In: Eigenthalet, G., et al. (eds.) Contributions to General Algebra, pp. 397–415. Wien, Hölder (1985)
[23] Wille, R.: Lattices in data analysis: how to draw them with a computer. In: Rival, I. (ed.) Algorithms and order, pp. 33–58. Kluwer Academic Publisher, Dordrecht (1989)

Approximations in Concept Lattices*

Christian Meschke

Institut für Algebra, TU Dresden, Germany
Christian.Meschke@tu-dresden.de

Abstract. Motivated by Rough Set Theory we describe an interval arithmetic on complete lattices. Lattice elements get approximated by *approximations* which are pairs consisting of a lower and an upper approximation. The approximations form a complete lattice again. We describe these lattices of approximations by formal contexts. Furthermore, we interpret the result for concept lattices as restricting the scope to a subcontext of *interesting* objects and attributes.

1 Introduction

Given a large, possibly infinite formal context (G, M, I) one usually has to handle a very large number of concepts which often yields to overloaded, unreadable order diagrams of the concept lattice. One practicable way to solve this problem is to use *nested line diagrams*. To build such a nested line diagram one splits the attribute set M into two not necessarily disjoint subsets M_1 and M_2, and embeds the concept lattice of (G, M, I) into the direct product of the concept lattices of the two subcontexts $(G, M_1, I \cap G \times M_1)$ and $(G, M_2, I \cap G \times M_2)$. The higher readability follows from the edge saving method to draw the direct product by copying the diagram of the second concept lattice into each node of the first one. Hence, when looking at the nested line diagram one has to look inside the big nodes when one is interested in the attributes from M_2, and one has to look at the *outer lattice* when one is interested in the attributes from M_1.

Just looking at the outer lattice of such a nested line diagram is equivalent to picking a subset $N \subseteq M$ of *interesting* attributes and looking at the concept lattice $\underline{\mathfrak{B}}(G, N, I \cap G \times N)$. What we are going to do is to additionaly pick a subset $H \subseteq G$ of *interesting* objects. Obviously, the restriction to the subcontext $(H, N, I \cap H \times N)$ yields to a smaller concept lattice, but one loses information about the interesting objects and attributes. An implication between interesting attributes that holds in (G, M, I) also holds in $(H, N, I \cap H \times N)$. But an implication $A \to B$ that holds in the subcontext does not necessarily have to hold in (G, M, I). One calls an object x *less general* than y in the context (G, M, I), if x has every attribute that y has. This gives rise to the object quasiorder defined by

$$x \sqsubseteq y \ :\Longleftrightarrow\ x^I \supseteq y^I.$$

If for $x, y \in H$ the object x is less general than y in (G, M, I), the object x is also less general than y in the extracted context $(H, N, I \cap H \times N)$. Hence, the

* Supported by the DFG research grant no. GA 216/10-1.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 104–123, 2010.

object quasiorder of (H, N, J) is a quasiorder extension of the object quasiorder of (G, M, I) restricted to H.

In summary we make the unsurprising observation that the restriction to the subcontext $(H, N, I \cap H \times N)$ yields to the negative side effect of losing information from (G, M, I) about the interesting objects and attributes. In order to avoid these problems we describe a familiar, but slightly more sophisticated method of restricting ourselves to interesting objects and attributes. It is based on so-called *approximations* in the concept lattice of (G, M, I). Thereby, approximations are pairs of concepts consisting of a *lower* and an *upper* approximation.

2 Approximations

In this section we describe so-called *approximations* in complete lattices. Let $\mathbf{L} = (L, \leq)$ be a complete lattice and let K be a kernel system and let C be a closure system in $\mathbf{L}$, i.e., for every $S \subseteq K$ and $T \subseteq C$ it holds that $\bigvee S \in K$ and $\bigwedge T \in C$. Furthermore, let $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ be the respective kernel[1] and closure operators. Hence, for $x \in L$ it holds that

$$\lfloor x \rfloor = \bigvee \{k \in K \mid k \leq x\} \quad \text{and} \quad \lceil x \rceil = \bigwedge \{c \in C \mid x \leq c\}.$$

We call $\lfloor x \rfloor$ the **lower approximation** of x and $\lceil x \rceil$ the **upper approximation** of x. Furthermore, we call the pair $(\lfloor x \rfloor, \lceil x \rceil)$ the **approximation generated by** x.

As an example from Rough Set Theory one can take for $\mathbf{L}$ the powerset lattice $(\mathfrak{P}(U), \subseteq)$ where the set U is the so-called *universe*. The approximations result from an equivalence relation $\sim$ on U which usually describes *indiscernibility* of objects. For $X \subseteq U$ the approximations are defined as follows:

$$\begin{aligned} \lfloor X \rfloor &:= \{u \in U \mid \forall v \sim u : \ v \in X\}, \\ \lceil X \rceil &:= \{u \in U \mid \exists v \sim u : \ v \in X\}. \end{aligned}$$

In this example, the kernel system equals the closure system. They consist exactly of those subsets of U that are the union of $\sim$ equivalence classes, the so-called *crisp* sets. It is well known that in this example the generated approximations $(\lfloor X \rfloor, \lceil X \rceil)$ (with $X \subseteq U$) form a lattice if one orders them by component-wise set inclusion.

In the case where $\mathbf{L}$ is a powerset lattice and where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are arbitrary kernel and closure operators, the generated approximations do not necessarily form a lattice. In [2] Ganter suggested to investigate the complete sublattice of $K \times C$ that is *generated* by the generated approximations $(\lfloor X \rfloor, \lceil X \rceil)$. Hence, this sublattice of approximations might contain pairs that are not generated by a subset of U, see also [5]. It is now an obvious step forward to investigate this approach for arbitrary complete lattices $\mathbf{L}$. Since K and C form complete

[1] Instead of using the terms *kernel system* and *kernel operator* it is quite common to use *interior system* and *interior operator* instead.

lattices, also $K \times C$ does. Thereby, infimum and supremum of a subset $\{(k_t, c_t) \in K \times C \mid t \in T\}$ are given by

$$\bigvee_{t\in T}(k_t, c_t) = (\bigvee_{t\in T} k_t, \lceil \bigvee_{t\in T} c_t \rceil),$$
$$\bigwedge_{t\in T}(k_t, c_t) = \Big(\lfloor \bigwedge_{t\in T} k_t \rfloor, \bigwedge_{t\in T} c_t\Big).$$

Definition 1. *The complete sublattice $\Gamma := \Gamma_{K,C}$ of $K \times C$ that is generated by the pairs $(\lfloor x \rfloor, \lceil x \rceil)$ with $x \in L$ is called the lattice of* **approximations**. *The kernel k is called the* **bottom** *and the closure c is called the* **top** *of an approximation (k, c).*

The notion of an approximation yields to an interval arithmetic on the complete lattice **L**. The following Section 3 investigates so-called *maximal approximations* and the role of *complete tolerance relations*, which yields to a better understanding and to a further generalisation of the approximations. In [2] the author gave a contextual representation of Γ for the special case where **L** is a power set lattice $(\mathfrak{P}(U), \subseteq)$. He therefore used so-called P-products which are the formal concept analytic way to describe subdirect products of complete lattices. In Section 5 we propose a similar contextual representation for our more general case and discuss properties of the so-called *concept approximations* and of its representing context. Section 4 provides the needed notions and propositions from Formal Concept Analysis.

3 Maximal Approximations

Even though the approximations (k, c) are defined to be special pairs of lattice elements, one automatically interprets them as intervals

$$[k, c] = \{x \in L \mid k \leq x \leq c\}.$$

in **L**. The reason for this interpretation is the simple fact that the bottom of an approximation is alway less or equal than the top. In other words one can say that $\Gamma_{K,C}$ is a subset of the order relation $\leq$. We say an approximation is **contained** in another one if its interpretation as an interval is a subset of the interval interpretation of the other approximation. Formally this **containment order** $\sqsubseteq$ on $\Gamma_{K,C}$ is defined by

$$(k_1, c_1) \sqsubseteq (k_2, c_2) \;:\Longleftrightarrow\; [k_1, c_1] \subseteq [k_2, c_2].$$

We call the maximal elements of the ordered set $(\Gamma_{K,C}, \sqsubseteq)$ the **maximal** approximations. Dually, we call an approximation **minimal** if it is a minimal element of $(\Gamma_{K,C}, \sqsubseteq)$. An approximation (k, c) with $k = c$ is called **crisp**. Obviously, crisp approximations are always minimal and minimal approximations are always generated by a lattice element. One can easily show using Zorn's lemma

that every approximation is contained in a maximal approximation. We will receive this result as a byproduct of the observation that the maximal approximations interpreted as intervals are the *blocks* of a *complete tolerance relation* on $\mathbf{L}$.

Definition 2 ([3]). *A binary relation $\Theta \subseteq L \times L$ is called a* **complete tolerance relation** *on* $\mathbf{L}$ *if it is reflexive, symmetric and compatible with suprema and infima, i.e., for which $x_t \Theta y_t$ $(t \in T)$ always implies*

$$(\bigwedge_{t\in T} x_t)\Theta(\bigwedge_{t\in T} y_t) \quad \text{and} \quad (\bigvee_{t\in T} x_t)\Theta(\bigvee_{t\in T} y_t).$$

Hence, a binary relation is a congruence relation iff it is transitive and a complete tolerance relation. If Θ is a complete tolerance relation on $\mathbf{L}$*, we define for $a \in L$*

$$a_\Theta := \bigwedge\{x \in L \mid a\Theta x\} \quad \text{and} \quad a^\Theta := \bigvee\{x \in L \mid a\Theta x\}.$$

The intervals $[a]_\Theta := [a_\Theta, (a_\Theta)^\Theta]$ are called the **blocks** *of Θ.*

Proposition 1. *The blocks of a complete tolerance relation Θ are precisely the maximal subsets X of L with $x\Theta y$ for all $x, y \in X$.*

Proof. See [3] Proposition 55. □

In our setting of a given kernel system K and a given closure system C in $\mathbf{L}$ we get a canonical tolerance relation $\Theta_{K,C}$ by

$$x\,\Theta_{K,C}\,y \;:\Longleftrightarrow\; \exists (k,c) \in \Gamma_{K,C} :\; \{x,y\} \subseteq [k,c].$$

The following propositions clarify the role of this tolerance relation $\Theta_{K,C}$.

Proposition 2. *For $x, y \in L$ it holds that*

$$x\,\Theta_{K,C}\,y \iff (\lfloor x \wedge y \rfloor, \lceil x \vee y \rceil) \in \Gamma_{K,C}.$$

Hence, for $k \in K$ and $c \in C$ with $k \leq c$ it holds that

$$k\,\Theta_{K,C}\,c \iff (k,c) \in \Gamma_{K,C}.$$

In other words one can write

$$\Gamma_{K,C} \;=\; (K \times C) \;\cap\; \leq \;\cap\; \Theta_{K,C}.$$

Proof. The second statement directly follows from the first. The backward direction of the first statement holds trivially. Let $x\,\Theta_{K,C}\,y$. Hence, there is an approximation (k,c) with $\{x,y\} \subseteq [k,c]$. Then it holds that

$$(\lfloor x \wedge y \rfloor, \lceil x \vee y \rceil) = \Big((k,c) \vee (\lfloor x \wedge y \rfloor, \lceil x \wedge y \rceil)\Big) \;\wedge\; (\lfloor x \vee y \rfloor, \lceil x \vee y \rceil).$$

□

Proposition 3. *The relation $\Theta_{K,C}$ is a complete tolerance relation. The blocks of $\Theta_{K,C}$ are precisely the intervals $[k, c]$ where (k, c) is a maximal approximation.*

Proof. Obviously $\Theta_{K,C}$ is symmetric and reflexive. Let $(x_t, y_t) \in \Theta_{K,C}$ for $t \in T$. Then for every $t \in T$ there is an approximation (k_t, c_t) with $\{x_t, y_t\} \subseteq [k_t, c_t]$. Since

$$\bigvee_{t\in T}(k_t, c_t) = \Big(\bigvee_{t\in T} k_t, \lceil \bigvee_{t\in T} c_t \rceil\Big)$$

is an approximation with

$$\Big\{\bigvee_{t\in T} x_t, \bigvee_{t\in T} y_t\Big\} \subseteq \Big[\bigvee_{t\in T} k_t, \lceil \bigvee_{t\in T} c_t \rceil\Big]$$

it follows $(\bigvee_{t\in T} x_t)\, \Theta_{K,C}\, (\bigvee_{t\in T} y_t)$. Dually one shows that $\Theta_{K,C}$ is compatible with the infimum. Let $X := [k, c]$ be an interval belonging to a maximal approximation (k, c) and let y be a lattice element fulfilling $x\, \Theta_{K,C}\, y$ for all $x \in X$. We show that $y \in X$ follows which implies by Proposition 1 that X is a block. From Proposition 2 we get that $(\lfloor k \wedge y \rfloor, \lceil k \vee y \rceil)$ is an approximation. Hence, also

$$(k, c) \vee (\lfloor k \wedge y \rfloor, \lceil k \vee y \rceil) = (k, \lceil c \vee \lceil k \vee y \rceil \rceil)$$

is an approximation as well. Since (k, c) is maximal we infer $c = \lceil c \vee \lceil k \vee y \rceil \rceil$ which implies $y \leq c$. Dually one shows $k \leq y$. Altogether we get $y \in X$. For the backward direction one takes a block $X = [k, c]$ of $\Theta_{K,C}$ and shows that (k, c) is a maximal approximation. With Proposition 2 one can argue that k is a kernel, that c is a closure and that (k, c) is an approximation. The maximality of (k, c) follows from Proposition 1. □

For a given complete tolerance relation the least elements of the blocks always form a kernel system. Dually, the greatest elements form a closure system. These two systems are isomorphic to each other, which allows to define a canonical order on the blocks and to *factorise* **L** (see [3]). Obviously, the kernel and closure system given by the blocks of $\Theta_{K,C}$ are subsystems of K and C, respectively. We call Θ a (K, C)**-tolerance** on **L** if it is a complete tolerance relation on **L** satisfying $x_\Theta \in K$ and $x^\Theta \in C$ for every $x \in L$. The (K,C)-tolerances form a closure system on $L \times L$, i.e., they are closed under intersections.

Proposition 4. *The relation $\Theta_{K,C}$ is the smallest (K, C)-tolerance.*

Proof. By Proposition 2 it suffices to show that every approximation is contained in every (K, C)-tolerance, i.e., $\Gamma_{K,C} \subseteq \Theta$ for every (K, C)-tolerance Θ. One proves this by first showing that every generated approximation $(\lfloor x \rfloor, \lceil x \rceil)$ belongs to Θ. Afterwards one easily shows that Θ is closed under the supremum and infimum as it is defined on $K \times C$. □

One can think of the tolerance $\Theta_{K,C}$ as having the role to ensure that bottom and top of an approximation do not differ too much. If one wants to define on its own what *not too much* means one can use the following generalisation of the notion of an approximation.

Definition 3. *Let* $\mathbf{L}$ *be a complete lattice, let* K *be a kernel system in* $\mathbf{L}$*, let* C *be a closure system in* $\mathbf{L}$ *and let* Θ *be a* (K, C)*-tolerance. We put*

$$\Gamma_{K,C,\Theta} := (K \times C) \cap \leq \cap \Theta$$

and call the pairs from $\Gamma_{K,C,\Theta}$ *the* (K, C, Θ)**-approximations**. *The notions* bottom, top, containment order, maximal, minimal *and* crisp *are defined analogously to the case of approximations where* $\Theta = \Theta_{K,C}$.

Proposition 5. $\Gamma_{K,C,\Theta}$ *is a complete sublattice of* $K \times C$*. For* $x, y \in L$ *it holds that*

$$x \Theta y \iff (\lfloor x \wedge y \rfloor, \lceil x \vee y \rceil) \in \Gamma_{K,C,\Theta}.$$

The blocks of Θ *are precisely the intervals* $[k, c]$ *where* (k, c) *is a maximal* (K, C, Θ)*-approximation.*

Proof. Let $(k_t, c_t) \in \Gamma_{K,C,\Theta}$ $(t \in T)$ and let

$$(k, c) := \bigvee_{t \in T} (k_t, c_t) = \Big(\bigvee_{t \in T} k_t, \lceil \bigvee_{t \in T} c_t \rceil \Big).$$

It obviously holds that $(k, c) \in K \times C \cap \leq$. Since Θ is a complete tolerance it follows $(\bigvee k_t, \bigvee c_t) \in \Theta$. Hence there is a block $[x, y]$ containing k and $\bigvee c_t$. Since furthermore Θ is a (K, C)-tolerance we infer $y \in C$ which implies $c \leq y$ and hence $(k, c) \in \Theta$. Dually one shows that $\Gamma_{K,C,\Theta}$ is closed under arbitrary infima. The rest can be shown similarly to the proofs of the Propositions 2 and 3. □

If $K = L$ and $C = L$, it follows $\Gamma_{K,C,\Theta} = \Theta \cap \leq$ for every complete tolerance relation Θ on $\mathbf{L}$. Thus, if one additionally chooses Θ to be the universal relation $L \times L$, it follows that the set of (K, C, Θ)-approximations equals the order relation of $\mathbf{L}$:

$$\Gamma_{K,C,\Theta} = \Gamma_{L,L,L \times L} = \leq .$$

4 Bonds and Block Relations

This section lists needed notions and propositions from Formal Concept Analysis. For a more detailed insight we refer the reader to [3]. Let $\mathbb{K} = (G, M, I)$ and $\mathbb{L} = (H, N, J)$ be formal contexts. A relation $B \subseteq G \times N$ is called a **bond** from $\mathbb{K}$ to $\mathbb{L}$ if every row of (G, N, B) is an intent of $\mathbb{L}$ and every column of (G, N, B) is an extent of $\mathbb{K}$. The set of all bonds from $\mathbb{K}$ to $\mathbb{L}$ is a closure system on $G \times N$. The respective closure operator is denoted by $(\cdot)^\beta$. Hence, for a relation $T \subseteq G \times N$ the closure T^β is the smallest bond from $\mathbb{K}$ to $\mathbb{L}$ that contains T.

Lemma 1. *For* $A \subseteq G$ *and* $B \subseteq N$ *it holds that*

$$A^{II} \times B^{JJ} \subseteq (A \times B)^\beta.$$

Proof. Let R be a bond from $\mathbb{K}$ to $\mathbb{L}$ with $A \times B \subseteq R$. It holds that $A^{II} \subseteq A^{RR}$ and $B \subseteq A^R$. Thus, it follows $B^{JJ} \subseteq A^{RJJ} = A^R$ and

$$A^{II} \times B^{JJ} \;\subseteq\; A^{RR} \times A^R \;\subseteq\; R.$$

□

The complete tolerance relations discussed in Section 3 have special bonds as its contextual counterpart, the so-called *block relations*. A relation $J \subseteq G \times M$ is called a **block relation** of the formal context (G, M, I) if it is a self-bond that contains I, i.e., if J is a bond from (G, M, I) to (G, M, I) with $I \subseteq J$.

Proposition 6. *The lattice of all block relations of (G, M, I) is isomorphic to the lattice of all complete tolerance relations on the concept lattice $\underline{\mathfrak{B}}(G, M, I)$. The map κ assigning to any complete tolerance relation Θ the block relation defined by*

$$g\kappa(\Theta)m \;:\Longleftrightarrow\; \gamma g \Theta(\gamma g \wedge \mu m) \quad (\Longleftrightarrow\; (\gamma g \vee \mu m)\Theta\mu m))$$

is an isomorphism. Conversely,

$$(A, B)\kappa^{-1}(J)(C, D) \;\Longleftrightarrow\; A \times D \cup C \times B \subseteq J$$

yields the complete tolerance to a block relation J.

Proof. See [3] Theorem 15. □

Corollary 1. *For a set U the lattice of all tolerance relations on the power set lattice $(\mathfrak{P}(U), \subseteq)$ is isomorphic to the lattice itself. The map τ assigning to any subset $X \subseteq U$ the complete tolerance relation $\tau(X)$ defined by*

$$(A, B) \in \tau(X) \;:\Longleftrightarrow\; A \cap X = B \cap X$$

is an isomorphism. Hence, every complete tolerance on a power set lattice is a congruence.

Proof. Follows from Proposition 6 since the block relations of the context $(U, U, \neq)$ are precisely the relations J_X with $X \subseteq U$ where

$$J_X := \{(x, y) \in U \times U \mid x = y \text{ implies } x \in X\}.$$

That τ is indeed an isomorphism is an elementary deduction from the definition of κ^{-1}. □

5 Concept Approximations

In this section we study the approximations from Section 2 on concept lattices. Obviously one can describe a kernel system K in a complete lattice by supremum-dense subsets of K, i.e., by subsets $T \subseteq K$ with

$$K \;=\; \{\bigvee S \mid S \subseteq T\}.$$

In the case of concept lattices we restrict ourselves to the kernel systems that are describable by object concepts. Since for a given concept it is always possible to extend the contexts object set in such a way that the concept is an object concept, this restriction is not a proper one regarding to the aim of describing arbitrary kernel systems in complete lattices. Dually we restrict ourself to closure systems given by subsets of the contexts attribute set.

Hence, we have the situation described in Section 1 where (G, M, I) is a **universal** context and where $(H, N, I \cap H \times N)$ is a subcontext called **selection**. Thereby the elements from H and from N are called the **interesting** objects and attributes, respectively. The subset $H \subseteq G$ yields to a kernel operator $\lfloor \cdot \rfloor_H$ on $\underline{\mathfrak{B}}(G, M, I)$ in the following canonical way:

$$\lfloor (A, B) \rfloor_H \ := \ ((A \cap H)^{II}, (A \cap H)^{I}).$$

Dually, $N \subseteq M$ yields to a closure operator via

$$\lceil (A, B) \rceil_N \ := \ ((B \cap N)^{I}, (B \cap N)^{II}).$$

In order to shorten our notations we define for sets A, B and for a relation R

$$R_{A,B} \ := \ R \cap A \times B.$$

Remark 1. For a concept $(A, B) \in \mathfrak{B}(G, M, I)$ the following three statements are equivalent:

(a) (A, B) is a kernel regarding to $\lfloor \cdot \rfloor_H$, i.e., $\lfloor (A, B) \rfloor_H = (A, B)$,
(b) B is an intent of $(H, M, I_{H,M})$,
(c) (A, B) is the supremum of object concepts γh with $h \in H$.

Dually, the following three statements are equivalent:

(d) (A, B) is a closure regarding to $\lceil \cdot \rceil_N$, i.e., $\lceil (A, B) \rceil_N = (A, B)$,
(e) A is an extent of $(G, N, I_{G,N})$,
(f) (A, B) is the infimum of attribute concepts μn with $n \in N$.

Hence, the kernel system K_H and the closure system C_N are the sets

$$K_H := \{(E^{II}, E^{I}) \mid E \subseteq H\} \text{ and}$$
$$C_N := \{(F^{I}, F^{II}) \mid F \subseteq N\}.$$

Ordered with the subconcept-superconcept order K_H and C_N are obviously isomorphic to the concept lattices of $(H, M, I_{H,M})$ and of $(G, N, I_{G,N})$, respectively. We denote the respective lattice of approximations with

$$\Gamma_{H,N} := \Gamma_{K_H, C_N}.$$

We call the pairs of concepts from $\Gamma_{H,N}$ **concept approximations**. For $E \subseteq H$ and $F \subseteq N$ we define

$$[\![E, F]\!] \ := \ ((E^{II}, E^{I}), (F^{I}, F^{II})).$$

Obviously, the pairs of the form $[\![E, F]\!]$ are exactly the pairs consisting of kernel in the first and a closure in the second component. It holds that

$$\begin{aligned} K_H \times C_N &= \{[\![E, F]\!] \mid E \subseteq H \text{ and } F \subseteq N\} \\ &= \{[\![E, F]\!] \mid E \in \mathrm{Ext}(H, M, I_{H,M}) \text{ and } F \in \mathrm{Int}(G, N, I_{G,N})\}, \end{aligned}$$

and for $E_t \in \mathrm{Ext}(H, M, I_{H,M})$ and $F_t \in \mathrm{Int}(G, N, I_{G,N})$ it holds that

$$\begin{aligned} \bigwedge_{t\in T} [\![E_t, F_t]\!] &= \Big[\!\Big[\bigcap_{t\in T} E_t, \big(\bigcup_{t\in T} F_t\big)^{II} \cap N \Big]\!\Big], \\ \bigvee_{t\in T} [\![E_t, F_t]\!] &= \Big[\!\Big[\big(\bigcup_{t\in T} E_t\big)^{II} \cap H, \bigcap_{t\in T} F_t \Big]\!\Big]. \end{aligned}$$

But which pairs of the form $[\![E, F]\!]$ are concept approximations? The bottom of a concept approximation $[\![E, F]\!]$ is a subconcept of the top, i.e, it holds that

$$(E^{II}, E^I) \leq (F^I, F^{II}).$$

This is equivalent to (E, F) being a preconcept, i.e., $E \times F \subseteq I$. But not all pairs $[\![E, F]\!]$ where (E, F) is a preconcept are concept approximations. Analogously to the approximations on complete lattices where certain complete tolerance relations played an important role, it will be certain block relations that play that role for the concept approximations. We call a relation J with $I \subseteq J \subseteq G \times M$ a (H, N)**-block relation** if it is a bond from $(G, N, I_{G,N})$ to $(H, M, I_{H,M})$. Hence, (H, N)-block relation are always block relations and the classical block relations are precisely the (G, M)-block relations.

Proposition 7. *The lattice of all (H, N)-block relations is isomorphic to the lattice of all (K_H, C_N)-tolerances on the concept lattice $\underline{\mathfrak{B}}(G, M, I)$. The map κ assigning to any (K_H, C_N)-tolerance Θ the (H, N)-block relation defined by*

$$g\kappa(\Theta)m \;:\Longleftrightarrow\; \gamma g \Theta(\gamma g \wedge \mu m) \quad (\Longleftrightarrow\; (\gamma g \vee \mu m)\Theta\mu m))$$

is an isomorphism. Conversely,

$$(A, B)\kappa^{-1}(J)(C, D) \iff A \times D \cup C \times B \subseteq J$$

yields the (K_H, C_N)-tolerance to a (H, N)-block relation J. For $A \subseteq G$ and $B \subseteq M$ the pair (A, B) is a concept of (G, M, J) if and only if $[(B^I, B), (A, A^I)]$ is a block of $\kappa^{-1}(J)$.

Proof. Obviously Proposition 7 is a mild generalisation of Proposition 6 and we just have to show that κ and κ^{-1} are well-defined. Let Θ be a (K_H, C_N)-tolerance and let $J := \kappa(\Theta)$ be the corresponding block relation. By [3] Corollary 57 the blocks of J are the intervals of the form $[(B^I, B), (A, A^I)]$ where (A, B) is a concept of (G, M, J). Hence, we get $(B^I, B) \in K_H$ and $(A, A^I) \in C_H$. By Remark 1 we get that B is an intent of $(H, M, I_{H,M})$ and that A is an extent of $(G, N, I_{G,N})$ for every $(A, B) \in \mathfrak{B}(G, M, J)$. Hence, J is a (H, N)-block relation.

Let now J be a (H,N)-block relation, let $\Theta := \kappa^{-1}(J)$ be the corresponding complete tolerance relation and let $(A,B) \in \mathfrak{B}(G,M,I)$. Then

$$(C,D) \;:=\; (A,B)^{\Theta}$$

is the greatest concept from $\underline{\mathfrak{B}}(G,M,I)$ with $A \times D \cup C \times B \subseteq J$ which is equivalent to $A \times D \subseteq J$ and $C \times D \subseteq J$. The first condition holds trivially since (C,D) is a superconcept of (A,B) and hence $A \subseteq C = D^I \subseteq D^J$. Thus (C,D) is the greatest superconcept of (A,B) satisfying the second condition $C \times B \subseteq J$, which directly yields to $C = B^J$. Hence, $(C,D) = (B^J, B^{JI})$ is a closure from C_N because $B^J \in \mathrm{Ext}(G,N,I_{H,N})$. Dually one shows that $(A,B)_\Theta \in K_H$. The rest follows from [3] Corollary 57. □

Lemma 2. *The relation $R \subseteq G \times M$ defined by*

$$R \;:=\; \bigcup_{(A,B)\in\mathfrak{B}(G,M,I)} (B \cap N)^I \times (A \cap H)^I$$

satisfies $I \subseteq R$ and $I^\beta = R^\beta$, where $(\cdot)^\beta$ denotes the bond closure operator for bonds from $(G,N,I_{G,N})$ to $(H,M,I_{H,M})$. Hence, R^β is the smallest (H,N)-block relation.

Proof. For $(g,m) \in I$ there is some $(A,B) \in \mathfrak{B}(G,M,I)$ with $(g,m) \in A \times B$. From $g \in A = B^I \subseteq (B \cap N)^I$ and $m \in B = A^I \subseteq (A \cap H)^I$ it follows $(g,m) \in R$.

Let T be a bond with $I \subseteq T$. We show $R \subseteq T$: For every $(A,B) \in \mathfrak{B}(G,M,I)$ it holds that

$$\begin{aligned}(B \cap N)^I \times (A \cap H)^I \;&=\; (A^I \cap N)^I \times (B^I \cap H)^I \\ &=\; A^{I_{G,N} I_{G,N}} \times B^{I_{H,M} I_{H,M}} \\ &\subseteq\; (A \times B)^\beta \\ &\subseteq\; I^\beta \;\subseteq\; T^\beta \;=\; T.\end{aligned}$$

Thereby the first inclusion follows from Lemma 1. Hence, a bond contains I iff it contains R. □

	M	N
H	$I_{H,M}$	$I_{H,N}$
G	I^β	$I_{G,N}$

Fig. 1. The context $\mathbb{A}_{H,N}$. Thereby I^β denotes the smallest bond from $(G,N,I_{G,N})$ to $(H,M,I_{H,M})$ containing I. For technical reasons we have to think of G and H as being replaced by disjoint copies. Analogously for M and N.

Theorem 1. *$\Gamma_{H,N}$ is isomorphic to the concept lattice of the context $\mathbb{A}_{H,N}$ displayed in Figure 1. An isomorphism is given by*

$$\begin{aligned}\varphi : \underline{\mathfrak{B}}(\mathbb{A}_{H,N}) &\longrightarrow \Gamma_{H,N}\\ (A,B) &\longmapsto [\![A \cap H, B \cap N]\!].\end{aligned}$$

A pair of concepts $[\![E,F]\!]$ where $E \subseteq H$ and $F \subseteq N$ is a concept approximation if and only if $E \times F \subseteq I$ and $F^I \times E^I \subseteq I^\beta$.

Proof. One can prove this Theorem 1 by showing that $\mathbb{A}_{H,N}$ is the P-fusion of the two P-contexts $((H, M, I_{H,M}), \alpha_H)$ and $((G, N, I_{G,N}), \alpha_G)$, where $P := \mathfrak{B}(G, M, I)$,

$$\alpha_H(A,B) := (A \cap H, (A \cap H)^I) \quad \text{and} \quad \alpha_N(A,B) := ((B \cap N)^I, B \cap N).$$

For this approach one needs Lemma 2. For details regarding P-fusions and P-contexts see [3]. We leave out the details of the proof since Theorem 1 is a special case of Theorem 2. □

It turns out that the bonds I^β and $I_{H,N}$ correspond one-to-one to the maximal and to the minimal concept approximations, respectively. In order to describe this we have to refresh two basic notions from Formal Concept Analysis. The context $(H, N, I_{H,N})$ is called a **dense** subcontext of (G, M, I) if $\gamma[H]$ is $\bigvee$-dense and $\mu[N]$ is $\bigwedge$-dense in $\underline{\mathfrak{B}}(G, M, I)$. The context $(H, N, I_{H,N})$ is called a **compatible** subcontext of (G, M, I) if the pair $(A \cap H, B \cap N)$ is a concept of $(H, N, I_{H,N})$ for every concept (A, B) of (G, M, I).

Proposition 8. *The maximal concept approximations are precisely the pairs of the form*

$$\big((B^I, B), (A, A^I)\big) = [\![B^I \cap H, A^I \cap N]\!]$$

where $(A, B) \in \mathfrak{B}(G, M, I^\beta)$. The mapping $(A, B) \mapsto \big((B^I, B), (A, A^I)\big)$ is an order-embedding of $\underline{\mathfrak{B}}(G, M, I^\beta)$ into $\Gamma_{H,N}$. It is an isomorphism if and only if the selection $(H, N, I_{H,N})$ is a dense subcontext of (G, M, I).

Proof. The mentioned equivalence follows from [3] Corollary 57 (3.). Hence, the mapping is well-defined. That it is an order-embedding is elementary: for $(A_i, B_i) \in (G, M, I^\beta)$ it holds that

$$(A_1, B_1) \leq (A_2, B_2) \iff \big((B_1^I, B_1), (A_1, A_1^I)\big) \leq \big((B_2^I, B_2), (A_2, A_2^I)\big).$$

The rest is obvious, since $(H, N, I_{H,N})$ is dense in (G, M, I) iff $K_H = C_N = \mathfrak{B}(G, M, I)$ holds. □

Proposition 9. *For every $(E, F) \in \mathfrak{B}(H, N, I_{H,N})$ the pair $[\![E, F]\!]$ of concepts is a concept approximation. The approximations of the form $[\![E, F]\!]$ where (E, F) is a concept of $(H, N, I_{H,N})$ are precisely the minimal concept approximations. Furthermore, an approximation $[\![E, F]\!]$ is crisp if and only if $(F^I, E^I) \in \mathfrak{B}(G, M, I)$.*

Proof. Let $(E,F) \in \mathfrak{B}(H,N,I_{H,N})$. Using Theorem 1 it suffices to show that $E \times F \subseteq I$ and $F^I \subseteq E^I \subseteq I^\beta$. The first item obviously holds. The second follows from Lemma 1:

$$F^I \times E^I = (E^I \cap N)^I \times (F^I \cap H)^I = E^{I_{G,N} I_{G,N}} \times F^{I_{H,M} I_{H,M}} \subseteq (E \times F)^\beta \subseteq I^\beta.$$

Let $[\![E,F]\!]$ be a concept approximation. W.l.o.g. we assume $E \in \mathrm{Ext}(H,M,I_{H,M})$ and $F \in \mathrm{Int}(G,N,I_{G,N})$. If $[\![E,F]\!]$ is not minimal, there is an approximation $[\![Q,R]\!]$ with

$$(E^{II},E^I) \;\leq\; (Q^{II},Q^I) \;\leq\; (R^I,R^{II}) \;\leq\; (F^I,F^{II})$$

where at most one of the two outer inequations is a proper $<$. Hence, it follows $E \subsetneq Q$ or $F \subsetneq R$ which implies $E \times F \subsetneq Q \times R \subseteq I_{H,N}$. Thus, (E,F) is not a concept of $(H,N,I_{H,N})$. Let us otherwise suppose that $[\![E,F]\!]$ is minimal and that $(E,F) \notin \mathfrak{B}(H,N,I_{H,N})$. Then there is a concept $(Q,R) \in \mathfrak{B}(H,N,I_{H,N})$ with $E \times F \subsetneq Q \times R$. We know from above that $[\![Q,R]\!]$ is an approximation. It holds that $Q^I \subseteq E^I$ and $R^I \subseteq F^I$. Equality of the first subset relationship implies

$$E \;=\; E^{I_{H,M} I_{H,M}} \;=\; E^{II} \cap H \;=\; Q^{II} \cap H \;=\; Q.$$

Dually, the equality $F^I = R^I$ implies $F = R$. Hence, $[\![Q,R]\!]$ is a concept approximation that is properly contained in $[\![E,F]\!]$. But this contradicts the minimality of $[\![E,F]\!]$. The characterisation of the crisp concept approximations directly follows from $[\![E,F]\!] = \big((E^{II},E^I),(F^I,F^{II})\big)$. □

Proposition 10. *The subcontext $(H,N,I_{H,N})$ is dense in $\mathbb{A}_{H,N}$ if and only if it is a compatible subcontext of (G,M,I).*

Proof. Lemma 2 from [4] says that $(H,N,I_{H,N})$ is dense in $\mathbb{A}_{H,N}$ iff the following three equations hold:

(i) $\mathrm{Ext}(H,M,I_{H,M}) = \mathrm{Ext}(H,N,I_{H,N})$,
(ii) $\mathrm{Int}(G,N,I_{G,N}) = \mathrm{Int}(H,N,I_{H,N})$, and
(iii) $I^\beta = \bigcup\{F^I \times E^I \mid (E,F) \in \mathfrak{B}(H,N,I_{H,N})\}$.

Let $(H,N,I_{H,N})$ be a compatible subcontext of (G,M,I). Then (i) and (ii) obviously hold. Furthermore, for the relation R from *our* Lemma 2 it holds that

$$R = \bigcup_{(E,F) \in \mathfrak{B}(H,N,I_{H,N})} F^I \times E^I.$$

If we show that R already is a bond, it follows (iii). For $m \in M$ it holds that

$$m^R \;=\; \bigcup\{F^I \mid (E,F) \in \mathfrak{B}(H,N,I_{H,N}) \text{ and } m \in E^I\} \;=\; (m^I \cap H)^{I_{G,N} I_{G,N}}.$$

The second equality follows from the fact that $(m^I \cap H, (m^I \cap H)^I \cap N)$ is a concept of $(H,N,I_{H,N})$ with the property that for every $(E,F) \in \mathfrak{B}(H,N,I_{H,N})$ with $m \in E^I$ it holds that

$$F^I \subseteq ((m^I \cap H)^I \cap N)^I = (m^I \cap H)^{I_{G,N} I_{G,N}}.$$

Dually one shows that g^R is an intent of $(H, M, I_{H,M})$. Hence, R is indeed a bond. Let now $(H, N, I_{H,N})$ be a subcontext fulfilling (i), (ii) and (iii). We show that $(H, N, I_{H,N})$ is a compatible subcontext of (G, M, I) by applying Proposition 35 from [3]. Analogously to the previously mentioned one gets from (iii) that $m^{I^\beta} = (m^I \cap H)^{I_{G,N} I_{G,N}}$. For $n \in N$ it follows

$$n^I \subseteq n^{I^\beta} = ((n^I \cap H)^I \cap N)^I \subseteq (n^{II} \cap N)^I = n^I,$$

which yields to $n^{I^\beta} = n^I$. Let $h \in H$ and $m \in M$ with $(h, m) \notin I$. Then by (i) there is an attribute $n \in N$ with $m^I \cap H \subseteq m^I \cap H$ and $(h, m) \notin I$. This implies

$$m^I \subseteq m^{I^\beta} = (m^I \cap H)^{I_{G,N} I_{G,N}} \subseteq (n^I \cap H)^{I_{G,N} I_{G,N}} = n^I.$$

Dually one can prove that $(H, N, I_{H,N})$ also fulfills the second condition of [3] Proposition 35. □

Proposition 11. *There is a natural embedding of* $\underline{\mathfrak{B}}(H, N, I_{H,N})$ *into* $\Gamma_{H,N}$. *The mapping*

$$\begin{aligned} \psi : \underline{\mathfrak{B}}(H, N, I_{H,N}) &\longrightarrow \Gamma_{H,N} \\ (E, F) &\longmapsto [\![E, F]\!] \end{aligned}$$

is an order embedding. Furthermore, ψ *is an isomorphism if and only if* $(H, N, I_{H,N})$ *is a compatible subcontext of* (G, M, I).

Proof. That ψ is well-defined follows from Proposition 9. Obviously ψ is order-preserving. That it is also order-reversing is elementary: for two concepts (E_1, F_1) and (E_2, F_2) of $(H, N, I_{H,N})$ with $\psi(E_1, F_1) \leq \psi(E_2, F_2)$ it follows $F_1^I \subseteq F_2^I$ which implies $E_1 = F_1^I \cap H \subseteq F_2^I \cap H = E_2$. Thus it follows $(E_1, F_1) \leq (E_2, F_2)$. The inverse isomorphism of φ from Theorem 1 maps every concept approximation $[\![E, F]\!]$ where w.l.o.g. E is an extent of $(H, M, I_{H,M})$ and F is an intent of $(G, N, I_{G,N})$ to the concept

$$\varphi^{-1}([\![E, F]\!]) := (E \uplus F^I, E^I \uplus F)$$

of $\mathbb{A}_{H,N}$. Hence, in order to finish our proof it suffices to show by Proposition 10 that $(H, N, I_{H,N})$ is a dense subcontext of $\mathbb{A}_{H,N}$ if and only if the mapping

$$\begin{aligned} \chi : \underline{\mathfrak{B}}(H, N, I_{H,N}) &\longrightarrow \underline{\mathfrak{B}}(\mathbb{A}_{H,N}) \\ (E, F) &\longmapsto (E \uplus F^I, E^I \uplus F) \end{aligned}$$

is surjective. Let $(H, N, I_{H,N})$ be dense in $\mathbb{A}_{H,N}$ and let $(E \uplus F^I, E^I \uplus F)$ be an arbitrary concept of $\mathbb{A}_{H,N}$. We have to show that (E, F) is a concept of $(H, N, I_{H,N})$. Let $\boxplus$ denote the incidence relation of $\mathbb{A}_{H,N}$. Since $(H, N, I_{H,N})$ is dense it follows

$$E \uplus F^I = (E \uplus F^I)^{\boxplus\boxplus} = E^{\boxplus\boxplus},$$

which implies

$$E^I \uplus F \;=\; (E \uplus F^I)^{\boxplus} \;=\; E^{\boxplus} \;=\; E^I \uplus E^{I_{H,N}}.$$

Hence, it follows $F = E^{I_{H,N}}$. Dually one shows $F^{I_{H,N}} = E$. Let now χ be surjective and let (A, B) be a concept of $\mathbb{A}_{H,N}$. Then there is a concept (E, F) of $(H, N, I_{H,N})$ with $(A, B) = (E \uplus F^I, E^I \uplus F)$. It follows that

$$(A \cap H)^{\boxplus} \;=\; E^{\boxplus} \;=\; E^I \uplus E^{I_{H,N}} \;=\; E^I \uplus\; F \;=\; B \;=\; A^{\boxplus}.$$

Dually one shows $(B \cap N)^{\boxplus} = B^{\boxplus}$. Hence, by [3] Propostion 39 $(H, N, I_{H,N})$ is a dense subcontext of $\mathbb{A}_{H,N}$. □

In the following we answer the question on how to integrate the further generalised (K, C, Θ)-approximations from Section 3 in the previously described contextual representation. As Theorem 2 will show, the obvious answer is to replace the block relation I^β by arbitrary (H, N)-block relations.

Definition 4. *Let J be a (H, N)-block relation and let $\Theta_J := \kappa^{-1}(J)$ be the corresponding (K_H, C_N)-tolerance (see Proposition 7). We put*

$$\Gamma_{H,N,J} := \Gamma_{K_C, H_N, \Theta_J}$$

and call the pairs from $\Gamma_{H,N,J}$ the (H, N, J)**-approximations**.

Increasing the block relation J obviously yields to greater but viewer maximal approximations, which again yields to increasing $\Gamma_{H,N,J}$.

	M	N
H	$I_{H,M}$	$I_{H,N}$
G	J	$I_{G,N}$

Fig. 2. The context $\mathbb{A}_{H,N,J}$, where J is a (H, N)-block relation

Theorem 2. *$\Gamma_{H,N,J}$ is isomorphic to the concept lattice of the context $\mathbb{A}_{H,N,J}$ displayed in Figure 2. An isomorphism is given by*

$$\begin{aligned} \varphi : \underline{\mathfrak{B}}(\mathbb{A}_{H,N,J}) &\longrightarrow \Gamma_{H,N,J} \\ (A, B) &\longmapsto [\![A \cap H, B \cap N]\!] \end{aligned}$$

The inverse isomorphism is the mapping that maps every (H, N, J)-approximation $[\![E, F]\!]$ where w.l.o.g. $E \in Ext(H, M, I_{H,M})$ and $F \in Int(G, N, I_{G,N})$ to

$$\varphi^{-1}([\![E, F]\!]) := (E \uplus F^I, E^I \uplus F).$$

	♢	♡	♠	♣	7	8	9	Q	K	10	A	♢J	♡J	♠J	♣J
♢7	×				×	×	×	×	×	×	×	×	×	×	×
♢8	×					×	×	×	×	×	×	×	×	×	×
♢9	×						×	×	×	×	×	×	×	×	×
♢Q	×							×	×	×	×	×	×	×	×
♢K	×								×	×	×	×	×	×	×
♢10	×									×	×	×	×	×	×
♢A	×										×	×	×	×	×
♡7		×			×	×	×	×	×	×	×	×	×	×	×
♡8		×				×	×	×	×	×	×	×	×	×	×
♡9		×					×	×	×	×	×	×	×	×	×
♡Q		×						×	×	×	×	×	×	×	×
♡K		×							×	×	×	×	×	×	×
♡10		×								×	×	×	×	×	×
♡A		×									×	×	×	×	×
♠7			×		×	×	×	×	×	×	×	×	×	×	×
♠8			×			×	×	×	×	×	×	×	×	×	×
♠9			×				×	×	×	×	×	×	×	×	×
♠Q			×					×	×	×	×	×	×	×	×
♠K			×						×	×	×	×	×	×	×
♠10			×							×	×	×	×	×	×
♠A			×								×	×	×	×	×
♣7				×	×	×	×	×	×	×	×	×	×	×	×
♣8				×		×	×	×	×	×	×	×	×	×	×
♣9				×			×	×	×	×	×	×	×	×	×
♣Q				×				×	×	×	×	×	×	×	×
♣K				×					×	×	×	×	×	×	×
♣10				×						×	×	×	×	×	×
♣A				×							×	×	×	×	×
♢J												×	×	×	×
♡J													×	×	×
♠J														×	×
♣J															×

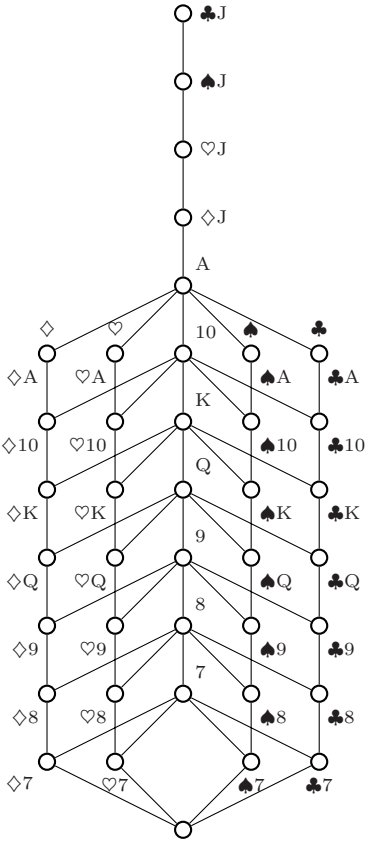


Fig. 3. The formal context (G, M, I) to the Skat example. Thereby Q stands for *queen*, K for *king*, A for *ace* and J for *jack*. The corresponding concept lattice has 40 concepts and is displayed on the right side. Note that the object concept of ♣J equals its attribute concept. The same holds for the other jacks. Hence, we do not have to label these concepts twice.

Proof. By [3] Theorem 32 the incidence relation $\boxplus$ of $\mathbb{A}_{H,N}$ is a closed subrelation in the sum context of $(H, M, I_{H,M})$ and $(G, N, I_{G,N})$. Hence, $\mathfrak{B}(\mathbb{A}_{H,N,J})$ is isomorphic to a complete sublattice $\mathfrak{S}$ of the direct product of $\underline{\mathfrak{B}}(H, M, I_{H,M})\,(\cong K_H)$ and of $\underline{\mathfrak{B}}(G, N, I_{G,N})\ (\cong\ C_N)$. An isomorphism is given via (see [3] Theorem 31)

$$\begin{aligned}\widehat{\varphi} : \underline{\mathfrak{B}}(\mathbb{A}_{H,N,J}) &\longrightarrow \mathfrak{S}\\ (A, B) &\longmapsto \big((A \cap H, B \cap M), (A \cap G, B \cap N)\big)\\ &= \big((A \cap H, (A \cap H)^I), ((B \cap N)^I, B \cap N)\big)\end{aligned}$$

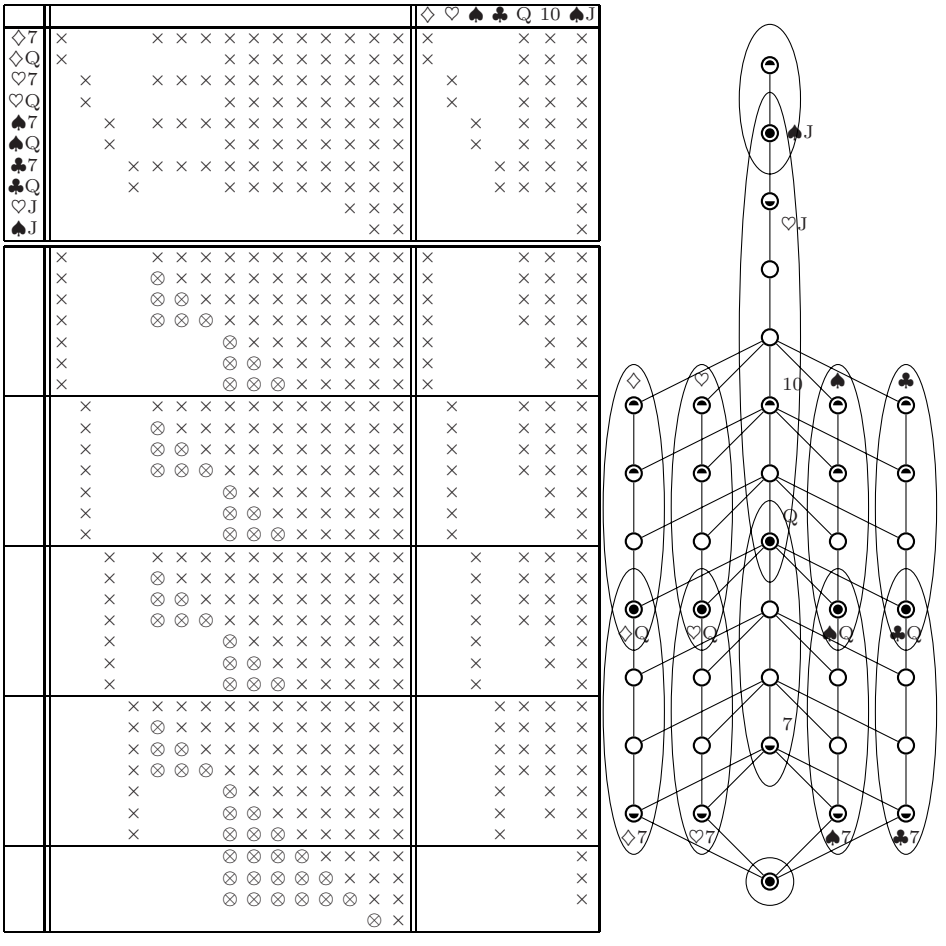


Fig. 4. The context $\mathbb{A}_{H,N}$ for our Skat example. We just labelled the interesting objects and attributes, which causes no problems since every concept approximation is of the form $[\![E, F]\!]$ with $E \subseteq H$ and $F \subseteq N$. The circled crosses $\otimes$ mark the pairs from I^β that do not belong to I. Note that the concept lattice of (G, M, I^β) is isomorphic to the lattice of all blocks; see Proposition 7. The right side shows a diagram of the concept lattice of (G, M, I), where just the interesting objects and attributes are labelled. The nodes having a filled lower half correspond to kernels. Closures are labelled by nodes that have a filled upper half. The 12 ellipse correspond to the blocks and hence to the maximal concept approximations.

Hence, for $E \in \mathrm{Ext}(H, M, I_{H,M})$ and $F \in \mathrm{Int}(G, N, I_{G,N})$ it holds that

$$\begin{aligned}\big((E, E^I), (F^I, F)\big) \in \mathfrak{S} &\iff (E \uplus F^I, E^I \uplus F) \in \mathfrak{B}(\mathbb{A}_{H,N,J}) \\ &\iff E \times F \subseteq I_{H,N} \text{ and } F^I \times E^I \subseteq J\end{aligned}$$

$$\begin{aligned} &\iff (E^{II}, E^I) \leq (F^I, F^{II}) \text{ and} \\ &\quad\ (E^{II}, E^I)\kappa^{-1}(J)(F^I, F^{II}) \\ &\iff [\![E, F]\!] \in \Gamma_{H,N,J}. \end{aligned}$$

Thereby the equivalence prior to the last one follows from Proposition 7 with the help of Lemma 1. Hence, $\mathfrak{S}$ is isomorphic to $\Gamma_{H,N,J}$ and the mapping φ indeed is an isomorphism. □

The attribute implications $A \to B$ with $A, B \subseteq N$ that hold in $\mathbb{A}_{H,N,J}$ are precisely the implications that hold in $(G, N, I_{G,N})$. Hence, these implications are precisely the attribute implications between interesting attributes that hold in the universal context (G, M, I). The dual statements holds for the interesting objects.

We close this section with a corollary from Proposition 7. It is a characterisation of the (K, C, Θ)-approximations for the case where $\mathbf{L}$ is a power set lattice.

Corollary 2. *Let $\mathcal{K}$ be a kernel system and $\mathcal{C}$ be a closure system* on *a set U. This means that $\mathcal{K}$ and $\mathcal{C}$ are a kernel system and a closure system* in *the power set lattice $\mathbf{L} := (\mathfrak{P}(U), \subseteq)$. Furthermore, let*

$$R := \{u \in U \mid \{u\} \in \mathcal{K} \text{ and } U \setminus \{u\} \in \mathcal{C}\}$$

be the set of so-called **robust** *elements. Then the following statements hold:*

(1) A pair $(X, Y) \in \mathcal{K} \times \mathcal{C}$ is an approximation iff X is a subset of Y and $Y \setminus X$ does not contain a robust element.

(2) Let $\tau(S)$ for $S \subseteq U$ be the complete congruence relation on $\mathbf{L}$ from Corollary 1 defined by

$$(A, B) \in \tau(S) \ :\iff A \cap S = B \cap S.$$

Then $\tau(S)$ is a $(\mathcal{K}, \mathcal{C})$-tolerance on $\mathbf{L}$ if and only if $S \subseteq R$. Since every complete tolerance relation on $\mathbf{L}$ is of the form $\tau(S)$ this characterises the $(\mathcal{K}, \mathcal{C})$-tolerances.

	♢	♡	♠	♣	Q	10	♠J
♢7	×				×	×	×
♢Q	×				×	×	×
♡7		×			×	×	×
♡Q		×			×	×	×
♠7			×		×	×	×
♠Q			×		×	×	×
♣7				×	×	×	×
♣Q				×	×	×	×
♡J							×
♠J							×

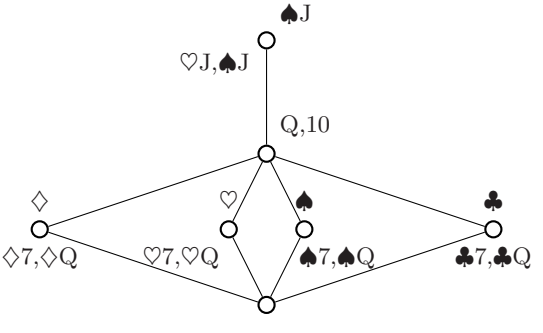


Fig. 5. The selection $(H, N, I_{H,N})$ and its concept lattice $\underline{\mathfrak{B}}(H, N, I_{H,N})$

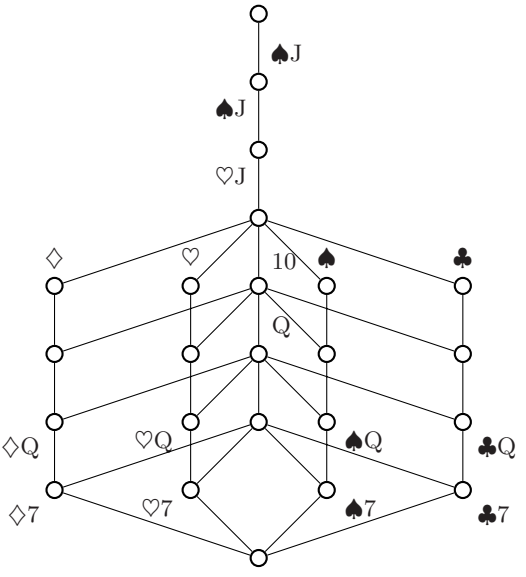


Fig. 6. The lattice of approximations $\Gamma_{H,N}$ for our Skat example. The corresponding context $\mathbb{A}_{H,N}$ is displayed in Figure 4. One reads the diagram as follows. Obviously the nodes represent the concept approximations $[\![E, F]\!]$. Similar to the reduced labelling of concept lattices, the elements from E are precisely the objects whose label can be found on the nodes below $[\![E, F]\!]$. Thereby *below* means that one can reach this node by going downwards along line paths in the diagram. Dually, the attributes from F are precisely the attributes labelling nodes above. As an example we take a look at the one unlabelled node at the very right. It represents the concept approximation $[\![E, F]\!] = \big((E^{II}, E^I), (F^I, F^{II})\big)$ with $E = \{\clubsuit 7, \clubsuit Q\}$ and $F = \{10, \clubsuit, \clubsuit J\}$.

(3) Let $S \subseteq R$. Then a pair $(X, Y) \in \mathcal{K} \times \mathcal{C}$ is a $(\mathcal{K}, \mathcal{C}, \tau(S))$-approximation iff $X \subseteq Y$ and $(Y \setminus X) \cap S = \emptyset$.

Proof. Statement (1) is from [5]. It is a special case of (3): By (2) $\tau(R)$ is the smallest $(\mathcal{K}, \mathcal{C})$-tolerance and hence by Proposition 4 it holds that $\Theta_{\mathcal{K},\mathcal{C}} = \tau(R)$. Statement (3) follows from (2) since for $X \subseteq Y$ the equivalences

$$(X, Y) \in \tau(S) \iff X \cap S \supseteq Y \cap S \iff X \subseteq Y \cap S \iff (Y \setminus X) \cap S = \emptyset$$

hold. For $S \subseteq U$ the blocks of $\tau(S)$ are precisely the intervals of the form

$$[T, (U \setminus S) \cup T] \;=\; [T, U \setminus (S \setminus T)],$$

where $T \subseteq S$. If $S \subseteq R$ it follows $T \in \mathcal{K}$ and $U \setminus T \in \mathcal{C}$ for every $T \subseteq S$. Hence, $\tau(S)$ is a $(\mathcal{K}, \mathcal{C})$-tolerance. If we otherise assume that $\tau(S)$ is a $(\mathcal{K}, \mathcal{C})$-tolerance, it follows that $\{x\} \in \mathcal{K}$ (put $T := \{x\}$) and $U \setminus \{x\} \in \mathcal{C}$ (put $T := S \setminus \{x\}$) for every $x \in S$. □

6 An Example

Our example is a toy example. It deals with the German card game Skat. Skat is a three player game that is played with a card deck consisting of 32 cards. These 32 cards are the objects of the context (G, M, I) displayed in Figure 3. The attributes and the incidence relation are chosen in such a way that the object quasiorder reflects the cards standard hierarchy. This means that for two cards x and y it holds

$$x^I \supseteq y^I$$

if and only if card y *beats* card x. With *standard* we mean that just the four jacks are trump. Hence, one can think – with one little exception – of (G, M, I) as a scaled context resulting from a many-valued context with the two attributes *suite* and *value*. Thereby the values *diamonds* ♢*, hearts* ♡*, spades* ♠ and *clubs* ♣ of the attribute *suit* are scaled nominally. The values of the second attribute *value* are scaled ordinally with the exception of the jacks. A jack is always trump which means that this card is above every non-jack in the cards hierarchy. The reader should note that furthermore the 10 beats the king of the same suit.

To start a Skat game each of the three players receives ten playing cards. The two remaining cards form the so-called *skat* and the player who wins the *bidding* process is allowed to use these two additional cards to build an improved combination of ten cards to play against his two opponents. From a players point of view the subset H of interesting cards might for instance be the ten cards he received at the beginning. Or maybe the interesting objects are the twelve cards he owns after winning the bidding for the skat. In order to receive a small lattice $\Gamma_{H,N}$ of approximations we chose the pretty regular set of playing cards

$$H := \{♢7, ♢Q, ♡7, ♡Q, ♠7, ♠Q, ♣7, ♣Q, ♡J, ♠J\}.$$

The choice of N might appear artificial, too. We took

$$N := \{♢, ♡, ♠, ♣, Q, 10, ♠J\}$$

which can be interpreted as coarsening the scale. The player might just be interested in the following questions: What is the suit of a given card? Is it weaker or equal than a queen or a 10 (of the same suit)? Is it weaker or equal than the jack of spades? The resulting selection $(H, N, I_{H,N})$ and its concept lattice is displayed in Figure 5. The Figures 4 and 6 show the context $\mathbb{A}_{H,N}$ and the corresponding lattice of approximations $\Gamma_{H,N}$.

Note that in our example the block relation I^β is relatively small, which yields to a relatively small number of concept approximations. It is for instance possible that the number of approximations exceeds the number of concepts of (G, M, I). But since the inequality

$$|\Gamma_{H,N}| \leq |\mathfrak{B}(H, M, I_{H,M})| \cdot |\mathfrak{B}(G, N, I_{G,N})|$$

trivially holds, it follows that relatively small subcontexts tend to result in lattices of approximations that are noticeably smaller than the concept lattice $\underline{\mathfrak{B}}(G, M, I)$.

7 Conclusion

We introduced and discussed approximations in complete lattices and described them via formal contexts. Furthermore, we interpreted the result as restricting the view from a formal context to a subcontext without losing implicational knowledge about the selected objects and attributes.

References

1. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press, Cambridge (2002)
2. Ganter, B.: Lattices of Rough Set Abstractions as P-Products. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 199–216. Springer, Heidelberg (2008)
3. Ganter, B., Wille, R.: Formal Concept Analysis – Mathematical Foundations. Springer, Heidelberg (1999)
4. Ganter, B.: Relational galois connections. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 1–17. Springer, Heidelberg (2007)
5. Meschke, C.: Robust Elements in Rough Set Abstractions. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 114–129. Springer, Heidelberg (2009)
6. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Dordrecht (1991)

Hardness of Enumerating Pseudo-intents in the Lectic Order

Felix Distel

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

Abstract. We investigate the complexity of enumerating pseudo-intents in the lectic order. We look at the following decision problem: Given a formal context and a set of n pseudo-intents determine whether they are the lectically first n pseudo-intents. We show that this problem is coNP-hard. We thereby show that there cannot be an algorithm with a good theoretical complexity for enumerating pseudo-intents in a lectic order. In a second part of the paper we introduce the notion of minimal pseudo-intents, i. e. pseudo-intents that do not strictly contain a pseudo-intent. We provide some complexity results about minimal pseudo-intents that are readily obtained from the previous result.

1 Introduction

The so-called stem base or Duquenne-Guigues Base from Formal Concept Analysis (FCA, [5]) plays an important rôle within FCA [6]. It has applications both within FCA as well as other fields such as Description Logics (DL) (in particular in knowledge base completion [1]). Therefore it is not surprising that it has been of major interest in the FCA community since its introduction.

In order to compute the Duquenne-Guigues Base of a formal context one must compute its pseudo-intents. The most well known algorithm for computing pseudo-intents is the *Next-Closure-Algorithm* [4]. It produces all concept intents and all pseudo-intents of a given formal context in a lexicographic order (called the lectic order). Another less well known algorithm has been introduced in 2007 [9,10]. It computes concept intents and pseudo-intents by starting with a set containing a single attribute and then incrementally adding attributes.

Both algorithms compute not only pseudo-intents but also concept intents. It is not difficult to see that the number of concept intents can be exponential in the number of pseudo-intents. As an example consider a series of contexts $\mathbb{K}_n = (G_n, M_n, I_n)$ where $M_n = \{1, \ldots, n\}$ and all subsets of M_n with cardinality $n-2$ are object intents. This context has $\frac{1}{2}n(n-1)$ objects and n attributes. The pseudo-intents of $\mathbb{K}_n$ are exactly the sets of cardinality $n-1$. All sets of cardinality less than $n-1$ are concept intents. This means that there are $2^n - n - 1$ concept intents while there are only n pseudo-intents. The case $n = 4$ is shown in Table 1.

This shows that there is a problem with the known algorithms for computing pseudo-intents. In many practical applications such as attribute exploration or

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 124–137, 2010.

Table 1. A Formal Context with 4 Pseudo-intents and $2^4 - 4 - 1$ Concept Intents

	1	2	3	4
g_1	X	X		
g_2	X		X	
g_3	X			X
g_4		X	X	
g_5		X		X
g_6			X	X

knowledge base completion one is not interested in concept intents but only in pseudo-intents. Yet, the above example shows that in the worst case the time needed to enumerate all pseudo-intents can be exponential in the size of the output, i. e. the number of pseudo-intents, when using one of the two known algorithms.

This raises the question whether it is theoretically possible to find more efficient algorithms for computing pseudo-intents. It is known that the number of pseudo-intents can be exponential in the size of the incidence relation of the context [7]. From this it immediately follows that there cannot be an algorithm that enumerates pseudo-intents in polynomial time in the size of the input (which would be the incidence relation).

For problems where the size of the output can be large in the size of the input other measures of complexity have been developped. One possibility is to take into account not only the size of the input, but also the size of the output. An algorithm is said to run in *output polynomial time* if it enumerates the solutions in time polynomial in the size of the input *and* the output. In previous work a relationship between the problem of enumerating pseudo-intents and the so-called transversal hypergraph problem (TransHyp, [2]) has been discovered. TransHyp is known to be in coNP but so far no hardness result has been shown. It is most likely not coNP-hard because it can be solved in $n^{o(\log n)}$ time [3]. It is also not known whether TransHyp is in P. It has been shown that pseudo-intents cannot be enumerated in output-polynomial time unless TransHyp is in P [11,12].

For someone who wants to apply attribute exploration in practice the most interesting measure of complexity is the delay between the computation of one pseudo-intent and the next. During this time the expert must wait unproductively for the next question to show up. With the known algorithms the delay can be exponential in the size of the input – and even in the size of the output. An enumeration algorithm is said to run with *polynomial delay* if the time between the enumeration of one solution and the next is polynomial in the size of the input.

The central question in this paper is whether it is possible to enumerate pseudo-intents in the lectic order with polynomial delay. We prove that the problem of checking whether a given set of n pseudo-intents is the set of the

lectically first n pseudo-intents is coNP-hard. We conclude, it is impossible to enumerate pseudo-intents in the lectic order with polynomial delay unless P = NP.

In a second part of the paper we look at a subclass of the class of pseudo-intents that we call minimal pseudo-intents. We show that it is tractable to check whether a given set is a minimal pseudo-intent. We also provide an algorithm that given a context will output a minimal pseudo-intent in polynomial time. We show that, surprisingly, it is not even possible to enumerate minimal pseudo-intents in output polynomial time unless P = NP.

2 Preliminaries

We briefly introduce the basic notions of formal concept analysis. A *formal context* is a tuple (G, M, I) where G and M are finite sets and $I \subseteq G \times M$ is a binary relation. The elements of G are called objects and elements of M are called attributes. FCA provides two *derivation operators* that are both denoted by $\cdot'$. For a set of objects $A \subseteq G$ one defines $A' = \{m \in M \mid \forall g \in A : gIm\}$. Analogously, for a set $B \subseteq M$ one defines $B' = \{g \in G \mid \forall m \in B : gIm\}$. Applying the two derivation operators successively yields the closure operators $\cdot''$. The $\cdot''$-closed subsets of M are called *concept intents*, while the $\cdot''$-closed subsets of G are called *concept extents*. A concept intent A is called *object intent* if it can be written as the closure of a singleton set $A = \{g\}'$, $g \in G$. Given a context (G, M, I) and a set $A \subseteq M$ one can check in time polynomial in the size of I and A whether A is a concept intent. The following Lemma is common knowledge in FCA.

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

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

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

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

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

In [6] a minimum cardinality base, which is called the *Duquenne-Guigues-Base*, has been introduced. The premises of the implications in the Duquenne-Guigues-Base are so-called pseudo-intents. $P \subseteq M$ is a *pseudo-intent* if P is not a concept intent and $Q'' \subseteq P$ holds for every pseudo-intent Q that is a proper subset of P.

The Duquenne-Guigues-Base consists of all implications $P \to P''$, where P is a pseudo-intent.

The well-known algorithm *Next-Closure* computes all pseudo-intents and concept intents in the lectic order [4]. The lectic order is defined as follows. Let a strict total order $<$ on the set M of attributes be given. Let $A, B \subseteq M$ be two sets of attributes. Define

$$A < B :\Leftrightarrow \exists i \in B - A : A \cap \{j \in M \mid j < i\} = B \cap \{j \in M \mid j < i\}.$$

If $A < B$ holds then we say that A *is lectically smaller than* B.

3 Enumerating Pseudo-intents in a Lectic Order

We have seen that the delay between the computation of two pseudo-intents is important. The two known algorithms do not have good theoretical properties. Both of them compute not only pseudo-intents, but also concept intents. For a given context the number of concept intents can be exponential in the number of pseudo-intents. That means that in the worst case, the algorithm would compute an exponential number of concept intents before the next pseudo-intent shows up. We ask whether it is possible to come up with an algorithm that behaves better. The answer is, if we require that the pseudo-intents be computed in the lectic order then there cannot be an algorithm with polynomial delay unless P = NP. We prove this by examining the following decision problem.

*Problem 1 (Lectically first pseudo-intents (*FirstPI*)). Input:* A formal context $\mathbb{K} = (G, M, I)$ and pseudo-intents $P_1, \ldots P_n$.
Question: Are $P_1, \ldots, P_n$ the n lectically first pseudo-intents of $\mathbb{K}$?

The dual problem to FirstPI would be "Given a formal context $\mathbb{K}$ and pseudo-intents $P_1, \ldots, P_n$ check if $P_1, \ldots, P_n$ are not the lectically first pseudo-intents of $\mathbb{K}$.". This problem can be characterized as follows.

Proposition 1. *$P_1, \ldots, P_n$ are not the n lectically first pseudo-intents of $\mathbb{K}$ iff there is a set $Q \subseteq M$ such that*

1. *Q is lectically smaller than P_j for some $j \in \{1, \ldots, n\}$, and*
2. *Q is not a concept intent, and*
3. *for all $i \in \{1, \ldots, n\}$ either $P_i \not\subseteq Q$ or $P_i'' \subseteq Q$.*

Proof. if: Because Q is not a concept intent there must be a pseudo-intent P of $\mathbb{K}$ such that $P \subseteq Q$ but $P'' \not\subseteq Q$. Because of 3 it holds that $P \notin \{P_1, \ldots, P_n\}$. P is lectically smaller than P_j because Q is lectically smaller than P_j and $P \subseteq Q$. Thus $P_1, \ldots, P_n$ are not the lectically smallest pseudo-intents of $\mathbb{K}$.

only if: Let P be a pseudo-intent that is lectically smaller than P_j, for some $j \in \{1, \ldots, n\}$ but not contained in $\{P_1, \ldots, P_n\}$. Then $Q = P$ satisfies the three conditions 1 to 3.

Lemma 2 (Containment in coNP). FirstPI *is in* coNP.

Proof. We show that the dual problem of FirstPI can be decided in non-deterministic polynomial time.

Whether a set $Q \subseteq M$ satisfies conditions 1 to 3 from Proposition 1 can be checked in time polynomial in the size of $\mathbb{K}$ and $P_1, \ldots, P_n$. In order to decide whether $P_1, \ldots, P_n$ are not the lectically first pseudo-intents of $\mathbb{K}$ one can non-deterministically guess a subset $Q \subseteq M$ and then check in polynomial time whether it satisfies 1 to 3. Hence the dual problem of FirstPI is in NP and thus FirstPI is in coNP.

For our hardness proof we use a reduction from the tautology problem, the prototypical coNP-complete problem.

*Problem 2 (*Tautology*). Input:* A boolean DNF-formula $f(p_1, \ldots, p_m) = (x_{11} \wedge \cdots \wedge x_{1l_1}) \vee \cdots \vee (x_{k1} \wedge \cdots \wedge x_{kl_k})$, where $x_{ij} \in \{p_1, \ldots, p_m\} \cup \{\neg p_1, \ldots, \neg p_m\}$. *Question:* Is f a tautology?

Tautology is coNP-complete, even with the restriction that f be in DNF. This is because f is a tautology iff $\neg f$ is unsatisfiable. If f is in DNF then $\neg f$ can be transformed to CNF in linear time. Checking if $\neg f$ is unsatisfiable is the dual problem of the Satisfiability Problem for boolean CNF formulae, which is, of course, NP-complete.

We prove that FirstPI is harder than Tautology by reduction. Let an instance f of Tautology be given. Let f be the DNF-formula $f(p_1, \ldots, p_m) = D_1 \vee \cdots \vee D_k$, where $D_i = (x_{i1} \wedge \cdots \wedge x_{il_i})$ and $x_{ij} \in \{p_1, \ldots, p_m\} \cup \{\neg p_1, \ldots, \neg p_m\}$ for all $i \in \{1, \ldots, k\}$ and all $j \in \{1, \ldots, l_i\}$. We define a context $\mathbb{K}$ as follows.

Let M be the set $M = \{\alpha_1, \ldots, \alpha_m, t_1, \ldots, t_m, f_1, \ldots f_m\}$. We define a total order $<$ on the elements of M as follows

$$\alpha_1 < \cdots < \alpha_m < t_1 < f_1 < \cdots < t_m < f_m.$$

For every $i \in \{1, \ldots, k\}$ define a set

$$\begin{aligned} A_i = M &- \{f_j \mid p_j \text{ occurs in } D_i \text{ as a positive literal}\} \\ &- \{t_j \mid p_j \text{ occurs in } D_i \text{ as a negative literal}\} \\ &- \{\alpha_j \mid p_j \text{ occurs in } D_i\} \end{aligned}$$

and furthermore for every $i \in \{1, \ldots, k\}$ and every $j \in \{1, \ldots, m\}$ let F_{ij} and T_{ij} be the sets $T_{ij} = A_i - \{f_j, \alpha_j\}$, $F_{ij} = A_i - \{t_j, \alpha_j\}$. Define the set of objects G to be $G = \{u_1, \ldots, u_{2m}\} \cup \{g_{T_{ij}} \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}\} \cup \{g_{F_{ij}} \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}\}$. The relation I is defined so that every object $g_{T_{ij}}$ has all the attributes that are contained in the set T_{ij} and analogously for $g_{F_{ij}}$. Furthermore I is such that every singleton set $\{t_i\}$ or $\{f_i\}$ occurs as the concept intent of some u_i. More formally, we define

$$\begin{aligned} I = &\{(u_{2i-1}, t_i) \mid i \in \{1, \ldots, m\}\} \cup \{(u_{2i}, f_i) \mid i \in \{1, \ldots, m\}\} \\ &\cup \{(g_{F_{ij}}, x) \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}, x \in F_{ij}\} \\ &\cup \{(g_{T_{ij}}, x) \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, m\}, x \in T_{ij}\}. \end{aligned}$$

Table 2. Context $\mathbb{K}$

	$\alpha_1 \ldots \alpha_m$	t_1	f_1	t_2	$f_2 \ldots t_m$	f_m
u_1		X				
$\vdots$				$\ldots$		
$\vdots$					$\ldots$	
u_{2m}						X
$g_{T_{11}}$		T_{11}				
$\vdots$		$\vdots$				
$g_{T_{1m}}$		T_{1m}				
$\vdots$		$\vdots$				
$g_{T_{k1}}$		T_{k1}				
$\vdots$		$\vdots$				
$g_{T_{km}}$		T_{km}				
$g_{F_{11}}$		F_{11}				
$\vdots$		$\vdots$				
$g_{F_{1m}}$		F_{1m}				
$\vdots$		$\vdots$				
$g_{F_{k1}}$		F_{k1}				
$\vdots$		$\vdots$				
$g_{F_{km}}$		F_{km}				

There are $2mk + 2m$ objects and $3m$ attributes, so the size of the context is $\mathcal{O}(m^2k + m^2)$. As sets $P_1, \ldots, P_m$ we define $P_i = \{t_i, f_i\}$ for all $i \in \{1, \ldots, m\}$.

The reduction may look complicated at first glance. The basic ideas in the design of the reduction are the following.

- Any assignment of truth values ϕ corresponds naturally to a subset of $\{t_1, f_1, \ldots, t_m, f_m\}$, namely the set

$$S_\phi := \{t_i \mid \phi(p_i) = \texttt{true}\} \cup \{f_i \mid \phi(p_i) = \texttt{false}\}. \tag{1}$$

- If ϕ makes D_i true then S_ϕ is a subset of A_i.
- If S_ϕ is a subset of A_i then S_ϕ is a concept intent.

To formally prove that this is a reduction from Tautology to FirstPI we need to show two things. First, we need to show that what we have obtained is really an instance of FirstPI and second, we need to show that f is a "Yes"-instance of Tautology if and only if $(\mathbb{K}, \{P_1, \ldots, P_m\})$ is a "Yes"-instance of FirstPI.

Lemma 3. $(\mathbb{K}, \{P_1, \ldots, P_m\})$ *is an instance of* FirstPI

Proof. All we have to show is that all P_i are pseudo-intents. Note that all strict subsets of P_i are concept intents in $\mathbb{K}$ (this is because all singleton subsets $\{t_i\}$

and $\{f_i\}$ are object intents of some u_i). To see that $\alpha_i \in P_i''$ and thus $P_i'' \neq P_i$ consider the sets A_r for $r \in \{1, \ldots, k\}$. If $P_i = \{t_i, f_i\} \subseteq A_r$ then by definition of A_r p_i does not occur in D_i. Therefore $\alpha_i \in A_r$. Let $s \in \{1, \ldots, m\}$ be an index of some set T_{rs}. If $P_i \subseteq T_{rs}$ then $P_i \subseteq A_r$ and $i \neq s$. Then $\alpha_i \in A_r$ holds and because $i \neq s$ it follows that $\alpha_i \in T_{rs} = A_r - \{f_s, \alpha_s\}$ Analogously $\alpha_i \in F_{rs}$ if $P_i \subseteq F_{rs}$. Therefore all objects that have all attributes from P_i also have α_i as an attribute and thus $\alpha_i \in P_i''$. Therefore $P_i'' \neq P_i$ must hold. Hence P_i is a pseudo-intent. Therefore $(\mathbb{K}, \{P_1, \ldots, P_m\})$ is an instance of FirstPI.

We show that $\mathbb{K}$ has a pseudo-intent that is lectically smaller than P_1 if and only if f is not a tautology. Let ϕ be an assignment that maps all p_i to a truth value in $\{\texttt{true}, \texttt{false}\}$. Let S_ϕ be defined as in (1). Note that S_ϕ contains exactly one element of $\{t_i, f_i\}$ for every $i \in \{1, \ldots, m\}$.

Lemma 4. *There is some $i \in \{1, \ldots, k\}$ for which $S_\phi \subseteq A_i$ if and only if $f(\phi(p_1), \ldots, \phi(p_m)) = \texttt{true}$.*

Proof. only-if: Let ϕ be such that $S_\phi \subseteq A_i$. Then by definition of A_i it holds that $f_j \notin S_\phi$, and thus $\phi(p_j) = \texttt{true}$, for all p_j that occur as positive literals in D_i (we have removed f_j from A_i). Analogously, $\phi(p_j) = \texttt{false}$ for all p_j that occur as negative literals. Hence all literals in D_i evaluate to `true` and therefore both D_i and the whole formula evaluate to `true`.

if: Now let ϕ be an assignment that makes f true. Since f is in DNF it evaluates to `true` iff at least one of the k implicants evaluates to true. Let D_i for some $i \in \{1, \ldots, k\}$ be an implicant that evaluates to true. Then $\phi(p_j) = \texttt{true}$ for all p_j that occur as positive literals D_i and $\phi(p_j) = \texttt{false}$ for all p_j that occur as negative literals in D_i. By definition of A_i and S_ϕ this implies $S_\phi \subseteq A_i$.

Lemma 5. *If $S_\phi \subseteq A_i$ then S_ϕ can be written as*

$$S_\phi = \bigcap_{\substack{j \in \{1,\ldots,m\} \\ \phi(p_j)=\texttt{true}}} T_{ij} \cap \bigcap_{\substack{j \in \{1,\ldots,m\} \\ \phi(p_j)=\texttt{false}}} F_{ij}$$

Proof. We denote the right-hand side of the above equation by R. By definition S_ϕ does not contain f_j if $\phi(p_j) = \texttt{true}$. Thus $S_\phi \subseteq A_i - \{f_j, \alpha_j\} = T_{ij}$ for all p_j for which $\phi(p_j) = \texttt{true}$. Likewise, $S_\phi \subseteq A_i - \{t_j, \alpha_j\} = F_{ij}$ for all p_j for which $\phi(p_j) = \texttt{false}$. Thus $S_\phi \subseteq R$. To prove the other inclusion consider some $x \in R$. For every $j \in \{1, \ldots, m\}$ it holds that $\alpha_j \notin F_{ij}$ and $\alpha_j \notin T_{ij}$. If $\phi(p_j) = \texttt{true}$ then $R \subseteq T_{ij}$, otherwise $R \subseteq F_{ij}$. So in either case $\alpha_j \notin R$. Therefore $x \neq \alpha_j$ holds for all $j \in \{1, \ldots, m\}$. Assume that $x = t_j$ for some j. Then $\phi(p_j) = \texttt{true}$ must hold, for otherwise R would be a subset of F_{ij} which does not contain t_j. Now $\phi(p_j) = \texttt{true}$ implies $x = t_j \in S_\phi$. The case $x = f_j$ for some j can be treated analogously. Thus for every $x \in R$ it holds that $x \in S_\phi$ and thus $R \subseteq S_\phi$. Hence $S_\phi = R$.

Lemma 6. *f is a tautology if and only if for all assignments ϕ the set S_ϕ is a concept intent of $\mathbb{K}$.*

Proof. Let us start by proving the *if*-direction. Assume that there is an assignment ϕ that makes f false. From Lemma 4 it follows that $S_\phi \not\subseteq A_i$ for all $i \in \{1, \ldots, k\}$. But then no object in G has all the attributes in S_ϕ because every object intent is either a singleton set or a subset of some A_i. Therefore $S_\phi'' = M$ and thus S_ϕ is not a concept intent. This contradicts the assumption and thus f must be a tautology.

For the *only if*-direction assume that there is some ϕ for which S_ϕ is not a concept intent. We know that the intersection of concept intents is also a concept intent. This implies in particular that S_ϕ cannot be written as the intersection of object intents. From Lemma 5 it follows that $S_\phi \not\subseteq A_i$ for all $i \in \{1, \ldots, k\}$. But then Lemma 4 shows that ϕ makes f false. This is a contradiction to the assumption that f is a tautology. Therefore S_ϕ must be a concept intent for all ϕ.

Lemma 7. *$P_1, \ldots, P_m$ are the lectically smallest pseudo-intents of $\mathbb{K}$ if and only if for all assignments ϕ the set S_ϕ is a concept intent in $\mathbb{K}$.*

Proof. *only-if*-direction: Assume that some S_ϕ is not a concept intent. Then S_ϕ has a subset $P \subseteq S_\phi$ which is a pseudo-intent. Obviously P is lectically smaller than P_1. Also P must be different from all the P_i because S_ϕ does not include any of the P_i. This is a contradiction to the assumption that $P_1, \ldots, P_m$ are the lectically smallest pseudo-intents of $\mathbb{K}$.

if-direction: Let $Q \subseteq M$ be a set of attributes that is lectically smaller than P_1. If Q would contain some α_i then it would be lectically larger than P_1. Therefore Q must be a subset of $\{t_1, f_1, \ldots, t_m, f_m\}$. If there is some $i \in \{1, \ldots, m\}$ such that $P_i \subseteq Q$ then $\alpha_i \in P_i'' - Q$ and thus $P_i'' \not\subseteq Q$. Therefore Q is not equal to P_i or a pseudo-intent. If $P_i \not\subseteq Q$ for all $i \in \{1, \ldots, m\}$ then define:

$$\phi_t(p_i) = \begin{cases} \texttt{true} & t_i \in Q \\ \texttt{false} & f_i \in Q \\ \texttt{true} & \text{otherwise} \end{cases} \qquad \phi_f(p_i) = \begin{cases} \texttt{true} & t_i \in Q \\ \texttt{false} & f_i \in Q \\ \texttt{false} & \text{otherwise} \end{cases}$$

Both ϕ_t and ϕ_f are well-defined since Q cannot contain both t_i and f_i for any i. With ϕ_t and ϕ_f defined as above it holds that $Q = S_{\phi_t} \cap S_{\phi_f}$. Since all S_ϕ are concept intents the intersection of S_{ϕ_t} and S_{ϕ_f} must also be a concept intent. Therefore Q cannot be a pseudo-intent.

Theorem 1 (Hardness of FirstPI). FirstPI *is* coNP*-hard.*

Proof. From Lemma 6 and Lemma 7 it follows that $P_1, \ldots, P_m$ are the lectically first pseudo-intents in $\mathbb{K}$ if and only if f is a tautology. Since the reduction can be done in polynomial time it follows that FirstPI is coNP-hard.

Corollary 1. FirstPI *is* coNP*-complete.*

What does this mean for the problem of enumerating pseudo-intents in the lectic order? Assume that there is an algorithm $\mathcal{A}$ that given a context enumerates its

pseudo-intents in the lectic order and with polynomial delay. That means that there is a polynomial $p(|G|, |M|)$ such that the delay between the computation of one pseudo-intent and the next is bounded by $p(|G|, |M|)$. Here $|M|$ denotes the number of attributes and $|G|$ denotes the number of objects in the context.

In order to solve FirstPI for an input $((G, M, I), \{P_1, \ldots, P_n\})$ we can construct a new algorithm $\mathcal{A}'$ from $\mathcal{A}$. $\mathcal{A}'$ lets $\mathcal{A}$ run for time $n \cdot p(|G|, |M|)$. After that time $\mathcal{A}$ will have computed the lectically first n pseudo-intents (and possibly some more, but these are not interesting). If these lectically first n pseudo-intents are identical to $P_1, \ldots, P_n$ then $\mathcal{A}'$ returns "Yes", otherwise it returns "No". The runtime of $\mathcal{A}'$ is bounded by $n \cdot p(|G|, |M|)$ and thus polynomial in the size of the input. Since FirstPI is coNP-hard, it cannot be solved in polynomial time unless P = NP.

Theorem 2. *Pseudo-intents cannot be enumerated in the lectical order with polynomial delay, unless* P = NP*.*

4 Minimal Pseudo-intents

4.1 Introducing Minimal Pseudo-intents

We say that P is a *minimal pseudo-intent of* $\mathbb{K}$ if P is a pseudo-intent of $\mathbb{K}$ and P does not contain any other pseudo-intent of $\mathbb{K}$. An equivalent definition is the following.

Definition 1 (Minimal Pseudo-Intent). *A* minimal pseudo-intent *of a context is a set $P \subseteq M$ such that*

- *P is not a concept intent, and*
- *every strict subset $S \subset P$ is a concept intent.*

Minimal pseudo-intents play a special rôle among the pseudo-intents of a given context. While the Duquenne-Guigues base is the most well known implication base, a given formal context $\mathbb{K}$ may have other implication bases. There may even be several implication bases with minimal cardinality. Minimal pseudo-intents are important since they have to occur as premises in all bases of a context, not just in the Duquenne-Guigues base.

To clarify this assume that $\mathcal{L}$ is a set of implications of the context $\mathbb{K}$. Let P be a minimal pseudo-intent of $\mathbb{K}$. Assume that $\mathcal{L}$ does not contain an implication whose left-hand side is P. Since all strict subsets of P are concept intents, there can be no implication $C \to D$ in $\mathcal{L}$ where $C \subseteq P$ but $D \not\subseteq P$. But then $P \to P''$ does not follow from $\mathcal{L}$ and thus $\mathcal{L}$ is not a concept intent.

Lemma 8. *If $\mathcal{L}$ is an implication base of a given context $\mathbb{K} = (G, M, I)$ and P is a minimal pseudo-intent of $\mathbb{K}$ then $\mathcal{L}$ contains an implication $P \to D$, $D \subseteq M$, whose premise is P.*

This shows that any algorithm that computes an implication base for a context inevitably has to compute all minimal pseudo-intent. This makes them an interesting subject for further research.

Given a context $\mathbb{K} = (G, M, I)$ and a set of attributes $P \subseteq M$ it is not hard to tell whether P is a minimal pseudo-intent. By definition, P is a minimal pseudo-intent if and only if it is not a concept intent and all its strict subsets are concept intents.

Lemma 9. *All strict subsets of P are concept intents if and only if all sets $P \setminus \{m\}$, $m \in P$, are concept intents.*

Proof. Assume that all sets of the form $P \setminus \{m\}$, $m \in P$, are concept intents. Let $S \subsetneq P$ be a strict subset. S can be written as the intersection

$$S = \bigcap_{m \in P \setminus S} (P \setminus \{m\}).$$

Since the intersection of concept intents is itself a concept intent S must be a concept intent. This proves the "if"-direction. The "only if"-direction is trivial.

Because of Lemma 9 we do not need to check for all strict subsets of P whether they are pseudo-intents. To test if P is a minimal pseudo-intent it suffices to perform $n + 1$ checks, namely checking whether each of the n sets $P \setminus \{m\}$, $m \in P$, is a concept intent and whether P itself is not a concept intent. Since checking whether a given set is a concept intent can be done in polynomial time it can be checked in polynomial time whether P is a minimal pseudo-intent. By comparison the best known algorithm to check whether a set P is a pseudo-intent runs in coNP [8,7].

4.2 Finding Minimal Pseudo-intents

Not only do the two algorithms Next Closure and Incremental Construction have an exponential delay in between the computation of one pseudo-intent and the next. One may even have to wait for some time exponential in the size of the context until even the first pseudo-intent is computed. This raises the question whether there can be an algorithm that finds at least one pseudo-intent in polynomial time. To the best knowledge of the author no such algorithm has yet been published. Lemma 9 gives us an idea for a minimal algorithm (Algorithm 1) that finds one minimal pseudo-intent in polynomial time.

The idea is the following. We start with the full attribute set M and check whether all its strict subsets are concept intents using Lemma 9. If they are all concept intents then the context has no pseudo-intents. If one of them is not a concept intent then it either contains a pseudo-intent or is a pseudo-intent itself. Then we continue by checking whether that subset has a subset that is not a concept intent and so on.

Lemma 10 (Soundness of Algorithm 1). *Let $\mathbb{K}$ be a context. If $\mathbb{K}$ has a pseudo-intent then Algorithm 1 returns a minimal pseudo-intent S upon termination.*

Algorithm 1. Algorithm for finding one minimal pseudo-intent

```
1: Input: 𝕂 = (G, M, I)
2: S := M
3: repeat
4:    finished := true
5:    for all m ∈ S do
6:       if S \ {m} is not a concept intent then
7:          S := S \ {m}
8:          finished := false
9:          exit for-loop
10:      end if
11:   end for
12: until finished
13: if S = M then
14:    print 𝕂 has no pseudo-intent
15: else
16:    return S
17: end if
```

Proof. Algorithm 1 remains in the **repeat**-loop until the variable `finished` is true. This means that upon termination for all $m \in S$ the set $S \setminus \{m\}$ is a concept intent. Otherwise `finished` would have been set to **false** in one of the iterations of the inner **for**-loop. It follows from Lemma 9 that all strict subsets of S are concept intents. If $S \neq M$ then S is itself not a concept intent (this has been checked in the previous iteration of the **repeat**-loop). Then S is a minimal pseudo-intent.

On the other hand if Algorithm 1 terminates with $S = M$ then both M and all of its subsets are concept intents. Thus $\mathbb{K}$ does not have any pseudo-intents.

Lemma 11 (Termination of Algorithm 1). *Algorithm 1 terminates after at most $|M|$ iterations of the repeat-loop. The total runtime is bounded by $O(|G| \cdot |M|^3)$.*

Proof. The algorithm starts with $S = M$. In each iteration of the repeat-loop one element is removed from S. The algorithm terminates if S is the empty set. Therefore it must terminate after at most $|M|$ iterations.

In each iteration of the repeat-loop the for-loop is entered at most $|S| < |M|$ times. Inside the for-loop the algorithm checks whether $S \setminus \{m\}$ is a concept intent. This check can be done in time of order $\mathcal{O}(|G||M|)$. Thus, the total runtime is bounded by $\mathcal{O}(|G||M| \cdot |M| \cdot |M|)$.

This shows that not only is it possible to check in polynomial time whether a given set of attributes is a minimal pseudo-intent, it is also possible to find an arbitrary minimal pseudo-intent in polynomial time. This raises hopes that it might be possible to compute at least the minimal pseudo-intents in polynomial time. Unfortunately, this is not the case, as we will see by examining the following problem.

*Problem 3 (All minimal pseudo-intents (*AllMPI*)). Input:* A formal context $\mathbb{K} = (G, M, I)$ and pseudo-intents $P_1, \ldots, P_n$.
Question: Are $P_1, \ldots, P_n$ all minimal pseudo-intents of $\mathbb{K}$?

Lemma 12 (Containment in coNP). AllMPI *is in* coNP.

Proof. We already know that checking whether a set $Q \subseteq M$ is a minimal pseudo-intent can be done in polynomial time (Lemma 9). So to decide whether $P_1, \ldots, P_n$ are not all the minimal pseudo-intents one can non-deterministically guess a set $Q \subseteq M$ such that $Q \notin \{P_1, \ldots, P_n\}$ and then check in polynomial time whether it is a minimal pseudo-intent. Thus the dual problem of AllMPI can be decided in non-deterministic polynomial time. Therefore AllMPI is in coNP.

Lemma 13 (Hardness of AllMPI). AllMPI *is* coNP*-hard.*

Proof. We use the same reduction as for Theorem 1. Given an instance of Tautology, i. e. a propositional formula f in disjunctive normal form, let $\mathbb{K}$ be the context from Table 2, constructed as in Section 3. We show that $P_1 = \{t_1, f_1\}$, $\ldots$, $P_m = \{t_m, f_m\}$, $P_{m+1} = \{\alpha_1\}$, $P_{2m} = \{\alpha_m\}$ are all the minimal pseudo-intents of $\mathbb{K}$ iff f is a tautology.

It has already been shown in the proof of Theorem 1 that $P_1, \ldots, P_m$ are minimal pseudo-intents. The empty set $\emptyset$ is a concept intent in $\mathbb{K}$. In $\mathbb{K}$ all objects intents g' for some $g \in G$ are such that $\alpha_i \in g'$ if and only if $\{t_i, f_i\} \subseteq g'$. Therefore, $\{t_i, f_i\}$ is contained in $\{\alpha_i\}'' = P''_{m+i}$. Thus $P_{m+1}, \ldots, P_{2m}$ are also minimal pseudo-intents. We can use the first three steps of the proof of Theorem 1.

We claim that $P_1, \ldots, P_{2m}$ are all minimal pseudo-intents of $\mathbb{K}$ iff for all assignments ϕ the set S_ϕ is a concept intent in $\mathbb{K}$. *only-if*: Assume that some S_ϕ is not a concept intent. Then S_ϕ must contain some minimal pseudo-intent $P \subseteq S_\phi$. The definition of S_ϕ (1) shows that S_ϕ does not contain α_i, and it contains either t_i or f_i but not both, for all $i \in \{1, \ldots, n\}$. Thus S_ϕ does not contain any of the $P_1, \ldots, P_{2m}$, and therefore it must be a new minimal pseudo-intent.

if: Let $Q \subseteq M$ be a set of attributes. If Q contains some α_i then Q cannot be a minimal pseudo-intent. Therefore Q is a subset of $\{t_1, f_1, \ldots, t_m, f_m\}$. In Lemma 7 it is shown that Q cannot be a pseudo-intent if the hypothesis holds. Thus there cannot be another minimal pseudo-intent.

Together with Lemma 6 this proves that $P_1, \ldots, P_{2m}$ are all minimal pseudo-intents of $\mathbb{K}$ iff f is a tautology. Thus AllMPI is coNP-hard.

Corollary 2. AllMPI *is* coNP*-complete.*

Corollary 3. *Given a context* $\mathbb{K}$ *the set of all minimal pseudo-intents of* $\mathbb{K}$ *cannot be computed in output-polynomial time unless* P = NP.

Proof. Assume that there was an algorithm $\mathcal{A}$ that takes $\mathbb{K}$ as its input and enumerates the set $\mathcal{P}$ of all minimal pseudo-intents in output-polynomial time. Let n be the number of pseudo-intents. This means that there is a polynomial

$p(|G|, |M|, |\mathcal{P}|)$ such that for all contexts $\mathbb{K} = (G, M, I)$ the runtime of $\mathcal{A}$ is bounded by $p(|G|, |M|, |\mathcal{P}|)$.

Then we can construct an algorithm $\mathcal{A}'$ that decides AllMPI as follows. Given a context $\mathbb{K}$ and a set of minimal pseudo-intents $\{P_1, \ldots, P_n\}$ $\mathcal{A}'$ runs $\mathcal{A}$ on $\mathbb{K}$ for at most $p(|G|, |M|, n)$ steps. If $\mathcal{A}$ does not terminate then there must be more than n minimal pseudo-intents, so $\mathcal{A}'$ return "No". If $\mathcal{A}$ terminates then $\mathcal{A}'$ compares the output of $\mathcal{A}$ to $\{P_1, \ldots, P_n\}$. If they are identical then $\mathcal{A}'$ return "Yes", otherwise "No". The runtime of $\mathcal{A}'$ is bounded by a polynomial in $|G|$, $|M|$ and $|\mathcal{P}|$.

Note that this does not yield a complexity result for the problem of computing all pseudo-intents. That is unless it can be shown that the total number of pseudo-intents of a context is bounded by a polynomial in the number of its minimal pseudo-intents. We conjecture that this is not the case.

5 Conclusion

In this work we have proved that the problem FirstPI of determining whether a given set of pseudo-intents is the set of lectically first pseudo-intents of a given context is coNP-complete. This helped us to prove that enumerating pseudo-intents in the lectic order is not tractable unless P = NP. From the results of previous work it only followed that enumerating pseudo-intents (in any order) is not tractable unless TransHyp is in P.

In the second section of the paper we have introduced minimal pseudo-intents. They play a special rôle because they occur in any implication base of a context, not only in the Duquenne-Guigues base. In many ways they are easier to handle than general pseudo-intents. For example we have shown that given a set of concept intents it is tractable to check whether it is a minimal pseudo-intent. Furthermore, one can find one minimal pseudo-intent in polynomial time. However, we have shown that the set of minimal pseudo-intents of a context cannot be computed in output polynomial time.

Future work. We conjecture that the lectic order is a source of complexity in the enumeration process. We therefore suggest that in order to develop efficient algorithms for computing pseudo-intents the FCA community should try to find alternatives to the lectic order. An idea might be incremental algorithms in the style of Obiedkov et al. [10]. Perhaps, it is also possible to compute all pseudo-intents by starting with the full set of attributes and then deleting attributes similar to Algorithm 1.

References

1. Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007). AAAI Press/The MIT Press (2007)

2. Eiter, T., Gottlob, G.: Hypergraph transversal computation and related problems. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 549–564. Springer, Heidelberg (2002)
3. Fredman, M.L., Khachiyan, L.: On the complexity of dualization of monotone disjunctive normal forms. Journal of Algorithms 21(3), 618–628 (1996)
4. Ganter, B.: Two basic algorithms in concept analysis. Preprint 831, Fachbereich Mathematik, TU Darmstadt, Darmstadt, Germany (1984)
5. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, New York (1997)
6. Guigues, J.-L., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Math. Sci. Humaines 95, 5–18 (1986)
7. Kuznetsov, S.O.: On the intractability of computing the Duquenne-Guigues base. Journal of Universal Computer Science 10(8), 927–933 (2004)
8. Kuznetsov, S.O.: Counting pseudo-intents and #P-completeness. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3874, pp. 306–308. Springer, Heidelberg (2006)
9. Kuznetsov, S.O., Obiedkov, S.A.: Algorithms for the construction of concept lattices and their diagram graphs. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 289–300. Springer, Heidelberg (2001)
10. Obiedkov, S., Duquenne, V.: Attribute-incremental construction of the canonical implication basis. Annals of Mathematics and Artificial Intelligence 49(1-4), 77–99 (2007)
11. Sertkaya, B.: Some computational problems related to pseudo-intents. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 130–145. Springer, Heidelberg (2009)
12. Sertkaya, B.: Towards the complexity of recognizing pseudo-intents. In: Dau, F., Rudolph, S. (eds.) ICCS 2009. LNCS, vol. 5662, pp. 284–292. Springer, Heidelberg (2009)

On Links between Concept Lattices and Related Complexity Problems

Mikhail A. Babin and Sergei O. Kuznetsov

State University Higher School of Economics,
Myasnitskaya 20, 101000 Moscow, Russia
mikleb@yandex.ru, skuznetsov@hse.ru

Abstract. Several notions of links between contexts – intensionally related concepts, shared intents, and bonds, as well as interrelations thereof – are considered. Algorithmic complexity of the problems related to respective closure operators are studied. The expression of bonds in terms of shared intents is given.

1 Introduction

In many applications one has to deal with data about a system changing in time. To analyze this dynamics one has to track similarities between different states of the system. The change of data makes the knowledge about the system change too. When one uses Formal Concept Analysis as the underlying mathematical model to describe this kind of dynamics, one needs to have tools for finding "similar" concepts in two concept lattices and "links" between similar concepts in them. For example, in [8] the study of dynamics of a scientific community is based on finding similar concepts of contexts that represent same community at different time. Similarity of concepts play important role in an earlier model of a network of concepts based on *multicontexts* [11]. In this paper we consider several notions that reflect links between concepts lattices, such as intensionally related concepts, shared intents, and bonds. We study algorithmic complexity of the problems related to computing closures and maximally related concepts. We also study the relation between shared intents and bonds.

The paper is organized as follows. In Section 2, we introduce basic definitions and discuss intensionally related concepts. In Section 3 we study shared intents of two contexts and study some important complexity problems related to shared intents. We also show that shared intents are related to bonds between contexts.

2 Intentionally Related Concepts

First we recall some basic notions of Formal Concept Analysis (FCA) [10,2].

Let G and M be sets, called the set of objects and the set of attributes, respectively. Let I be a relation $I \subseteq G \times M$ between objects and attributes: for $g \in G$, $m \in M$, gIm holds iff the object g has the attribute m. The triple

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 138–144, 2010.

$K = (G, M, I)$ is called a *(formal) context*. Formal contexts are naturally represented by cross tables, where a cross for a pair (g, m) means that this pair belongs to the relation I. If $A \subseteq G$, $B \subseteq M$ are arbitrary subsets, then the *Galois connection* is given by the following *derivation operators*:

$$A' := \{m \in M \mid gIm \text{ for all } g \in A\},$$
$$B' := \{g \in G \mid gIm \text{ for all } m \in B\}.$$

The pair (A, B), where $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$ is called a *(formal) concept (of the context K)* with *extent* A and *intent* B. For $g \in G$ and $m \in M$ the sets $\{g\}'$ and $\{m\}'$ are called *object intent* and *attribute extent*, respectively. The set of attributes B *is implied by the set of attributes* D, or an *implication* $D \to B$ holds, if all objects from G that have all attributes from the set D also have all attributes from the set B, i.e., $D' \subseteq B'$.

The operation $(\cdot)''$ is a closure operator [2], i.e., it is idempotent ($X'''' = X''$), extensive ($X \subseteq X''$), and monotone ($X \subseteq Y \Rightarrow X'' \subseteq Y''$). Sets $A \subseteq G$, $B \subseteq M$ are called *closed* if $A'' = A$ and $B'' = B$. Obviously, extents and intents are closed sets. Since the closed sets form a closure system or a Moore space [1], the set of all formal concepts of the context K forms a lattice, called a *concept lattice* and usually denoted by $\underline{\mathfrak{B}}(K)$ in FCA literature.

Let $K_1 = (G_1, M, I_1), K_2 = (G_2, M, I_2), \ldots, K_r = (G_r, M, I_r)$ be contexts with common attribute set M. Denote by $(\cdot)^i$ the derivation operator in context K_i. A tuple of r concepts $(A_1, B_1), (A_2, B_2), \ldots, (A_r, B_r))$ of corresponding contexts $K_1, K_2, \ldots, K_r$ are called *intentionally related* [8] if

$$(\bigcap_{1 \le i \le r} A_i)^{11} = A_1$$
$$(\bigcap_{1 \le i \le r} A_i)^{22} = A_2$$
$$\ldots$$
$$(\bigcap_{1 \le i \le r} A_i)^{rr} = A_r$$

So any intentionally related concepts are uniquely defined by the set $\bigcap_{1 \le i \le r} A_i$.

Consider an operator $(\cdot)^*$, defined as $X^* = X^{11} \cap X^{22} \cap \ldots \cap X^{rr}$ for $X \subseteq M$.

Proposition 1. *Let $K_1 = (G_1, M, I_1), K_2 = (G_2, M, I_2), \ldots, K_r = (G_r, M, I_r)$ be contexts with common attribute set M. Then the operator $(\cdot)^*$ has the following properties:*
(1) $(X^*)^{ii} = X^{ii}$, for any $X \subseteq M$ and $1 \le i \le r$.
(2) $(\cdot)^*$ is a closure operator.

Proof. (1) Indeed, $(X^*)^{ii} = (\bigcap_{1 \le j \le r} X^{jj})^{ii}$, since $X \subseteq X^{jj}$ for any $1 \le j \le r$ it follows that $X \subseteq \bigcap_{1 \le j \le r} X^{jj}$ and hence $X^{ii} \subseteq (X^*)^{ii}$. On the other hand $\bigcap_{1 \le j \le r} X^{jj} \subseteq X^{ii}$, therefore $(X^*)^{ii} \subseteq X^{ii}$.

(2) It is not hard to check that this operator is a closure operator:
1. $X \subseteq Y \Rightarrow X^{ii} \subseteq Y^{ii}$, for $1 \leq \mathrm{i} \leq \mathrm{r} \Rightarrow X^* \subseteq Y^*$ (monotony)
2. $X \subseteq X^*$(was proved above) (extensity)
3. $X^{**} = \bigcap_{1 \leq j \leq r} (X^*)^{jj} = \bigcap_{1 \leq j \leq r} X^{jj} = X^*$ (idempotency)
□

The situation is easily extended to the case where contexts $K_1, K_2, \ldots, K_r$ have different sets of attributes $M_1, M_2, \ldots, M_r$. One defines $M := \bigcup_{i=1}^{r} M_i$ and proceeds like above.

Having the closure operator $(\cdot)^*$, one can compute all intentionally related concepts by standard algorithms (Norris, Next Closure, Close-by-One, etc.).

3 Concepts with Shared Intents

As in the previous section, let $K_1 = (G_1, M, I_1), K_2 = (G_2, M, I_2), \ldots, K_r = (G_r, M, I_r)$ be contexts with common attribute set M, $(\cdot)^i$ denotes the derivation operator in K_i for $1 \leq i \leq k$. A set $A \subseteq M$ is called *shared intent* for contexts $K_1, K_2, \ldots, K_r$ if it is an intent for every context K_i, i.e. $A^{ii} = A$.

Since for any context the set of all its intents generates a closure system, the set of all shared intents also generates a closure system. Let us denote the corresponding closure operator by $(\cdot)^{\mathcal{S}}$.

Theorem 2. *The problem*

INPUT Formal contexts $K_1 = (G_1, M, I_1), K_2 = (G_2, M, I_2), \ldots, K_r = (G_r, M, I_r)$ with common attribute set M, and a set $X \subseteq M$.

OUTPUT The closure $X^{\mathcal{S}}$ of X.

can be solved in $O(|M| \sum_{1 \leq i \leq r} |G_i|)$ time.

Proof. Consider sets $S_i = \{g' \mid g \in G_i, X \subseteq g'\} \cup \{M\}$, $1 \leq i \leq r$. Denote by $\bigcap S_i = \bigcap_{A \in S_i} A$. We will keep invariant that for every $1 \leq i \leq r$, any shared intent that contains X can be obtained by intersection of some elements of S_i.

Suppose that there exists an attribute $m \in M$ such that for some $1 \leq i \leq r$ one has $m \in \bigcap S_i$. Then, since every shared intent that contains X can be obtained by intersection of some elements of S_i, every shared intent that contains X have to contain m. Hence, if for some $1 \leq j \leq r$ there is an element $A \in S_j$ which does not contain m, we can update S_j by removing this element while keeping the invariant. When no such removal can be done, we try to find another element $m \in M$ that is contained in all elements of some S_i, and so on. Since M is finite, at some step there is no such $m \in M$. This means that any element $m \in M$ either belongs to every element of any $S_i, 1 \leq i \leq r$ or it does not belong to some element of S_i for every $1 \leq i \leq r$. Hence $\bigcap S_1 = \bigcap S_2 = \ldots = \bigcap S_r$, $X \subseteq \bigcap S_1$ and $\bigcap S_1$ is contained in any shared intent that contains X i.e. $\bigcap S_1 = X^{\mathcal{S}}$.

```
GetClosure(X)
 1  answer ← X
 2  for i ← 1 to r
 3        do remove all rows from I[i] that do not contain X
 4  for i ← 1 to r
 5        do for m ← 1 to |M|
 6              do if not answer[m]
 7                    then if I[i][j][m] = TRUE for all 1 ≤ j ≤ |G_i|
 8                              then answer[m] ← TRUE
 9                                   push m in shared-attributes
10                              else  for each 1 ≤ j ≤ |G_i| such that not I[i][j][m]
11                                         do push (j, i) in not-in[m]
12                                            counter[i][m] ← counter[i][m] + 1
13  while shared-attributes not empty
14        do pop m from shared-attributes
15           while not-in[m] not empty
16                 do pop (object-index, context-index) from not-in[m]
17                    for i ← 1 to |M|
18                          do if not I[i][context-index][object-index]
19                                then counter[context-index][i] ←
                                  ← counter[context-index][i] −1
20                                     if counter[context-index][i] ≤ 0
                                       and not answer[i]
21                                        then answer[i] = TRUE
22                                             push i in shared-attributes
23  return answer
```

Fig. 1. Pseudocode of the algorithm for computing $(\cdot)^{\mathcal{S}}$

Here $I[i]$ is a binary table representation of relation I_i, $counter[i][m]$ is the number of objects g of G_i for which gI_im does not hold, *shared-attributes* and *not-in*$[m]$ can be implemented as *stacks* or as any other data structure, with operations "pop" any object from it and "push" any object in it in $O(1)$ time. □

The problem of finding a maximal cardinality closed set wrt. $(\cdot)'$ and $(\cdot)^*$ is trivially polynomial: one just checks all object intents and finds the largest one. In contrast to that, a similar problem of finding a maximal size shared intent different from M (in practical data analysis this may correspond to the largest similarity of two contexts) for operator $(\cdot)^{\mathcal{S}}$ is NP-complete, as shown in the following

Proposition 3. *The problem*

INPUT Two formal contexts $K_1 = (G_1, M, I_1), K_2 = (G_2, M, I_2)$, and integer $0 \leq k \leq |M|$.

QUESTION Does there exist a set X such that $X = X^{\mathcal{S}}$, $X \subset M$ and $|X| \geq k$?

is NP-complete.

Proof. By Theorem 2 $X = X^{\mathcal{S}}$ can be checked in polynomial time, so the problem is in NP. For proving NP-hardness we reduce a well-known NP-complete problem – *minimal set cover* (MSC) to this one. The MSC problem is formulated as follows [4]:

INPUT Finite set S, some set of its subsets $\mathcal{P} := \{S_1, S_2, \ldots, S_m\}$, $S_i \subseteq S$, and integer k.

QUESTION Does there exist a subset $\mathcal{T} \subseteq \mathcal{P}$ such that $\bigcup_{X \in \mathcal{T}} X = S$ and $|\mathcal{T}| \geq k$?

Consider an arbitrarily finite set $S = \{s_1, s_2, \ldots, s_n\}$, a set of its subsets $\mathcal{P} := \{S_1, S_2, \ldots, S_m\}$, $S_i \subseteq S$ for all $1 \leq i \leq m$, and an integer $0 \leq k \leq |\mathcal{P}|$. Let us define sets $M = \{m_1, m_2, \ldots, m_{|S|}, m_{|S|+1}, m_{|S|+2}, \ldots, m_{|S|+|\mathcal{P}|}\}$ and $G_1 = \{g_1^1, g_2^1, \ldots, g_{|\mathcal{P}|}^1\}$. Now construct context $K_1 = (G_1, M, I_1)$, where I_1 is defined as follows: $g_i^1 I_1 m_j$ for $j \leq |S|$ iff $s_j \notin S_i$ and $g_i^1 I_1 m_j$ for $j > |S|$ iff $j - |S| \neq i$. Then covers of S are in one-to-one correspondence with intents of K_1 that do not contain any $m_i \in M$ for $i \leq |S|$. Moreover, for any set cover of size N, the corresponding intent has size $|\mathcal{P}| - N$.

Now let us construct context $K_2 = (G_2, M, I_2)$, where $G_2 = \{g_1^2, g_2^2, \ldots, g_{|\mathcal{P}|}^2\}$, I_2 is defined as follows: for g_i^2, where $1 \leq i \leq |\mathcal{P}|$, $g_i' = M \setminus (S \cup \{m_{|S|+i}\})$. Obviously, the set of all intents of this context is exactly the set of all subsets M that are disjoint with S. Hence there is one-to-one correspondence between the set of shared intents of K_1 and K_2 excluding M, and set of all set covers. Moreover the minimal covers correspond to maximal shared intents, and maximal shared intents correspond to minimal covers. The reduction is proved, its polynomiality is obvious. □

It is interesting to know how large can be the context for the set of all shared intents? The answer is given by the following proposition:

Proposition 4. *There exist two contexts K_1 and K_2 such that the set of maximal (by inclusion) their shared intents is exponential in size of K_1 and K_2.*

Proof. Consider finite set $S = \{s_1, s_2, \ldots, s_{3n}\}$, and set of its subsets $\mathcal{P} = \bigcup_{0 \leq i \leq n-1} \{\{s_{3i+1}, s_{3i+2}\}, \{s_{3i+1}, s_{3i+3}\}, \{s_{3i+2}, s_{3i+3}\}\}$. There are 3^n minimal covers of S, since for any $0 \leq i \leq n-1$ the subset $\{s_{3i+1}, s_{3i+2}, s_{3i+3}\}$ can be covered only in three ways, using exactly 2 elements from $\mathcal{P}$. So, if we construct contexts K_1 and K_2 like in Proposition 3 according to such S and $\mathcal{P}$, then there are 3^n maximal (by cardinality, and hence by inclusion) shared intents of K_1 and K_2. □

Corollary. There exist two contexts K_1 and K_2 such that the minimal number of objects in the context representing $(\cdot)^{\mathcal{S}}$ is exponential in sizes of K_1 and K_2.

3.1 Bonds as Shared Intents

In [2] the following definition of a bond was given

Definition 5. *Let $K_1 = (G_1, M_1, I_1)$ and $K_2 = (G_2, M_2, I_2)$ be contexts. A relation $I \subseteq G_1 \times M_2$ is called a* bond *from $K_1 = (G_1, M_1, I_1)$ to $K_2 = (G_2, M_2, I_2)$ if m^I is extent of K_1 for any $m \in M_2$, and g^I is intent of K_2 for any $g \in G_1$.*

Recall some definitions from [2]. The *direct product* of contexts $K_1 = (G_1, M_1, I_1)$ and $K_2 = (G_2, M_2, I_2)$ is given by

$$K_1 \times K_2 := (G_1 \times G_2, M_1 \times M_2, \nabla)$$

$$\text{with } (g_1, g_2)\nabla(m_1, m_2) \Leftrightarrow g_1 I_1 m_1 \text{ or } g_2 I_2 m_2.$$

Contranominal scale N_S^c is the context $(S, S, \neq)$

Proposition 6. *A relation $B \subseteq G_1 \times M_2$ is a bond from context $K_1 = (G_1, M_1, I_1)$ to context $K_2 = (G_2, M_2, I_2)$ iff B is a shared intent of contexts $N^c_{G_1} \times K_2$ and $(K_1 \times N^c_{M_2})^d$.*

Proof. Let $B \subseteq G_1 \times M_2$ be a shared intent of $N^c_{G_1} \times K_2 = (G_1 \times G_2, G_1 \times M_2, \nabla^2)$ and $(K_1 \times N^c_{M_2})^d = (G_1 \times M_2, M_1 \times M_2, \nabla^1)^d$. Since B is an intent of $N^c_{G_1} \times K_2$, one has

$$B = \bigcap_{(u,h) \in B^{\nabla^2}} \{(g, m) \mid (u, h)\nabla^2(g, m)\}$$

$$= \bigcap_u \bigcap_{h \in H(u)} (\{(u, m) \mid hI_2 m\} \cup \{(g, m) \mid g \neq u\}),$$

where $(g, m) \in G_1 \times M_2$, $\bigcap_u$ is taken over all u such that $(u, h) \in B^{\nabla^2}$ for some h, and $H(u) = \{h \mid (u, h) \in B^{\nabla^2}\}$. Hence if $g = u$ for some u taking part in intersection $\bigcap_u$ we have $g^B = \bigcap_{h \in H(u)} h^{I_2}$, i.e. g^B is closed in K_2, and if $g \neq u$ for all u taking part in $\bigcap_u$ then $g^B = M_2$. Similarly, since B is intent of $(K_1 \times N^c_{M_2})^d$, we can prove that m^B is closed in context K_1 for any $m \in M_2$.

Let $B \subseteq G_1 \times M_2$ be a bond from context K_1 to context K_2. Then u^B is closed in K_2 for any $u \in G_1$. Denote $H(u) = (u^B)^{I_2}$, then $u^B = \bigcap_{h \in H(u)} h^{I_2}$. Consider

$$D = \bigcap_u \bigcap_{h \in H(u)} (\{(u, m) \mid hI_2 m\} \cup \{(g, m) \mid g \neq u\}).$$

Above we showed that this set is an intent of both $N^c_{G_1} \times K_2$ and $(K_1 \times N^c_{M_2})^d$, i.e., a shared intent of these contexts. Then $g^D = g^B$ for any $g \in G_1$ and $m^D = m^B$ for any $m \in M_2$, hence $D = B$, and B is a shared intent of $N^c_{G_1} \times K_2$ and $(K_1 \times N^c_{M_2})^d$. □

Corollary. The closure operator for bonds of contexts $K_1 = (G_1, M_1, I_1)$ and $K_2 = (G_2, M_2, I_2)$ can be computed in $O((|G_1| \cdot |G_2| + |M_1| \cdot |M_2|) \cdot |M_2||G_1|)$ time.

Proof. Apply Proposition 6 and Theorem 2.

Conclusion

We considered several definitions that relate concepts of different contexts, such as intentionally related concepts, shared intents, and bonds. The types of links between concepts of different contexts are important for the study of context dynamics. Algorithmic complexity of problems related to corresponding closure operators was studied. We showed that bonds may be described in terms of shared intents. As further research we would like to find an answer to the question raised in [9] on whether bonds have natural concise context representation.

Acknowledgments

The authors would like to thank Sergei Obiedkov for helpful discussions. The second author was supported by the project of the Russian Foundation for Basic Research, grant no. 08-07-92497-NTsNIL_a.

References

1. Birkhoff, G.: Lattice Theory. Amer. Math. Soc., Providence (1967)
2. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
3. Ganter, B.: Lattices of Rough Set Abstractions as P-Products. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 199–216. Springer, Heidelberg (2008)
4. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
5. Kuznetsov, S.O.: On Computing the Size of a Lattice and Related Decision Problems. Order 18(4), 313–321 (2001)
6. Kuznetsov, S.O.: On the Intractability of Computing the Duquenne-Guigues Base. Journal of Universal Computer Science 10(8), 927–933
7. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell. 14(2-3), 189–216 (2002)
8. Kuznetsov, S.O., Obiedkov, S.A., Roth, C.: Reducing the Representation Complexity of Lattice-Based Taxonomies. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 241–254. Springer, Heidelberg (2007)
9. Krötzsch, M., Malik, G.: The Tensor Product as a Lattice of Regular Galois Connections. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3874, pp. 89–104. Springer, Heidelberg (2006)
10. Wille, R.: Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts. In: Rival, I. (ed.) Ordered Sets, pp. 445–470. Reidel, Dordrecht (1982)
11. Wille, R.: Conceptual Structure of Multicontexts. In: Eklund, P., Mann, G.A., Ellis, G. (eds.) ICCS 1996. LNCS (LNAI), vol. 1115, pp. 23–39. Springer, Heidelberg (1996)

An Algorithm for Extracting Rare Concepts with Concise Intents

Yoshiaki Okubo and Makoto Haraguchi

Division of Computer Science
Graduate School of Information Science and Technology
Hokkaido University
N-14 W-9, Sapporo 060-0814, Japan
{yoshiaki,mh}@ist.hokudai.ac.jp

Abstract. This paper presents an algorithm for finding concepts (closures) with smaller supports. As suggested by the study of emerging patterns, contrast sets or crossover concepts, we regard less frequent and rare concepts.

However, we have several difficulties when we try to find concepts in those rare concepts. Firstly, there exist a large number of concepts closer to individual ones. Secondly, the lengths of intents become longer, involving many attributes at various levels of generality. Consequently, it becomes harder to understand what the concepts mean or represent.

In order to solve the above problems, we make a restriction on formation processes of concepts, where the formation is a flow of adding attributes to the present concepts already formed. The present concepts work as conditions for several candidate attributes to be added to them. Given such a present concept, we prohibit adding attributes strongly correlated with the present concept. In other words, we add attributes only when they contribute toward decreasing the supports of concepts to some extent. As a result, the detected concepts has lower supports and consist of only attributes directing at more specific concepts through the formation processes.

The algorithm is designed as a top-N closure enumerator using branch-and-bound pruning rules so that it can reach concepts with lower supports by avoiding useless combination of correlated attributes in a huge space of concepts. We experimentally show effectiveness of the algorithm and the conceptual clarity of detected concepts because of their shorter length in spite of their lower supports.

1 Introduction

In these decades, much attention is paid for the study of *Formal Concept Analysis* and closure enumeration [1,2]. Many effective enumeration engines, [4,5,6,7] for instance, have been developed to have frequent closures (intents of larger extents). All of these are concerned with relatively frequent closures. This is simply based on the idea that everything valuable is frequent to some extent and non-frequent closures are exceptional and minor ones and are therefore disregarded.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 145–160, 2010.

On the other hand, in the research area of data mining, namely association rule mining, rather less frequent and rare patterns in their occurrences, that may not necessarily be closures, are considered significant under some conditions. For instance, *emerging patterns* [8,9], and *contrast sets* [10,11] are concerned with less frequent and even rare patterns, given some conditions. More precisely, the conditions are given by the selection of databases from some family of databases, and the patterns to be detected are required to have lower support (frequency) in one database, DB_1, and higher support in another database DB_2. It is thus expected to have non-frequent patterns at least in one database DB_1. Namely, the study of emerging patterns [8,9] makes emphasis on the emergence of patterns in the sense that patterns may indicate some important changes when they appear with higher support in DB_2 than in DB_1.

In contrast to emerging patterns and contrast sets, the authors have considered an intent X over two intents, A and B, such that X is a common generalization (join) of smaller subconcepts of A and B, respectively, where the smallness of concepts is simply the smallness of extent size [12]. We called such an X a *crossover* of A and B. The intuition behind the definition is that, even when the corresponding extent is smaller and rare, it may show some hidden connection among major concepts from some unusual viewpoints, and is therefore worth examining. The implemented algorithm, which is designed based on a branch-and-bound strategy and is tested for an incident relation made from document-term data, succeeded to find "*term*" and "*condition*" concept over "*Commercial Banks*" concept and "*Visual Basic*" concept. Although most documents about commercial banks and Visual Basic do not have the attribute specified by "*term*" and "*condition*", at least their smaller set of documents are concerned with the same concept of "*term*" and "*condition*". Note that, as the corresponding parts of crossover concept are small in both "*Commercial Banks*" and "*Visual Basic*", the concept is never emerging patterns nor contrast set. However, we consider that there may exist some hidden and non-standard inter-concept relationship between the concepts.

This paper is also concerned with the detection of concepts with smaller and rare extents. However, it is an obvious general fact that intents become longer as the extents are smaller. According to our past experiments on crossover concepts, the length of intent of 11 documents (0.002 support in the incident relation of $11,000$ documents and $1,223$ attribute terms) was 111 where the extent size was tried to be maximized under the constraint that length of intent must be over 50. As is easily imagined, it is a hard task to check the meaning of concepts of over 100 attributes. We found its meaning by checking the original documents to see that the detected concept is subsumed by more general concept, "*terms and condition*", that also un-covers many objects in "*Commercial Bank*" and "*Visual Basic*".

From the above observation, we particularly consider in this paper the following standpoint and research goals:

General standpoint:
A concept consisting of *less number* of *more general* attributes is more understandable and preferable. At the same time, we require those concepts to have smaller support (less number of objects in their extents). We regard them as valuable concepts to be examined.

Development of strategy and algorithms to find them:
As the number of less frequent concepts is huge in general, standard enumeration approach will not work. We therefore direct our attention to optimization algorithms for maximizing/minimizing evaluation of extents/intents under some constraints about intents/extents.

Based on the above, we introduce in this paper a special class of concepts, *Concise and Rare Concepts* (*CRC* for short), and present a top-down search algorithm to find them effectively. The intuitive idea and the rough definition of *CRC*s are as follows:

- Any concept can be defined as a general concept satisfying some specific constraints. This can be considered as a standard top-down concept definition style [13].
- So, we arrange attributes of an intent A by the decreasing order of their support. Any intent is thus viewed as an ordered set of attributes.
- Then, we interpret a given concept as a definitional context based on which less frequent attributes are added to form more specific concept working as a next context.
- When the supports of the corresponding two concepts change a little, we ignore the added attribute for the specific concept, since it can be approximated by the general concept. This is a one-step formation from general to specific ones.
- The degree of admissible support change is given by a parameter, δ. *CRC* is then defined as the last concept of a chain of one-step formations starting from the closure of empty attribute set.
- In order to exclude too many individual concepts, we set the second parameter, ϵ , and we impose a constraint that any significant *CRC* must be maximal in the partially ordered set of concepts with their supports less than ϵ.

Even when we consider only significant *CRC* in the above sense, there we still have a lot of *CRC*s, depending on the parameters. So we restrict the number of output *CRC* by a monotone evaluation function for intents of *CRC*s. Each intent is evaluated by the minimum support of its attributes.

Preference of *CRC*s:
As the minimum support of attributes of *CRC* is greater, we regard the *CRC* more general and preferable in our linear ordering based on the evaluation function.

Final output *CRC*s:
We compute only Top-N *CRC*s with respect to the evaluation, where N is a natural number given as the third parameter.

The algorithm for the above can enjoy a branch-and-bound pruning technique, and is quite effective even for incident relation of over $10,000$ objects and $1,000$ attributes.

In the literature [16], a framework of *Rare Itemset Mining* has been discussed. The rareness in the framework is also defined based on support of itemsets, where an itemset corresponds to a set of attributes in our case. However, since they do not take any semantics of rare itemsets into account, we are forced to extract many meaningless itemsets according to the framework. Furthermore, the algorithm presented in [16] can be viewed as an extension of `APRIORI` [3], it would not be able to work with reasonable computation time for large scale data.

A framework for finding formal concepts satisfying given constraints has also been discussed in D-Miner. We can, however, impose constraints based on only sizes of extents and intents. Any semantics of concepts is out of their interests. Moreover, as will be mentioned later, the computational performance of their algorithm is insufficient for extracting rare concepts we try to detect, even if we do not care any semantics.

With the help of closed itemset miners, e.g. `LCM` [7] and `D-Miner` [15], therefore, we can obtain formal concepts. Such systems are, however, not helpful for finding our rare concepts. Given a parameter *minsup*, they can usually enumerate all *frequent* closed itemsets in the sense that their supports are greater than or equal to *minsup*. If we try to find our rare concepts with those miners, we have to give a quite small value of *minsup*. As a well known fact, under such a low *minsup*, frequent closed itemset miners are quite useless because the number of frequent closures is enormous.

Although the notion and the method presented in this paper is just the beginning, the authors believe that it can be extended so as to find crossover concepts, contrast sets and emerging patterns more effectively. In Concluding Remarks, we discuss some possible approaches towards this direction.

Now in Section 2, we present our problem more precisely, and present our algorithm in details in Section 3. In Section 4, some experimental results will be shown to demonstrate the ability of our algorithm. In fact, it succeeds to find a *CRC* with 9 general terms, including "*contact*", "*UK*", "*application*", "*2001*" and "*update*" with low frequency. But, "*contact*", "*UK*", "*application*" and "*update*" for another year except "*2001*" is not outputted. This will motivate us to check if something unusual things about "*contact*", "*UK*", "*application*" and "*update*" in "*2001*" happen in the data set.

2 Concise Rare Concepts

As has discussed just above, we especially try to extract rare formal concepts in the sense that their extents are relatively small. The authors expect that such rare concepts might provide us a chance to perceive a hidden relationship between major (large) concepts.

Let (G, M, I) be a formal context. For a set of attributes $A \subseteq M$, the number of objects in G which are concerned with A is called the *support* of A and is denoted by $support(A)$, that is, $support(A) = |A'|$.

Our *rareness* of formal concept is simply evaluated by size of the extent, that is, its support.

Definition 1. (Rareness of Formal Concept)
Let (X, A) be a formal concept and ϵ a *rareness threshold*. The concept is said to be ϵ-*rare* if and only if $|X| = support(A) \leq \epsilon$. ■

From the theoretical property of formal concepts, as extents become smaller or rare, their intents tend to be larger. However, if the intent of a concept consists of many attributes, it is difficult to adequately interpret a meaning of the concept. In particularly, if the intent includes many *specific* attributes, the meaning is even unclearer. It might be practically worthless to extract such concepts with unclear meanings. From this point of view, the authors consider that concepts to be extracted must own *concise* intents in the sense that they consist of *general* attributes for high interpretability. In order to formalize such concepts, we define a notion of *generality* of intents.

The support of an attribute is regarded as a measure of *generality* of the attribute. Based on the notion of support, we define generality of intents.

Definition 2. (Generality of Intents)
Let (X, A) be a formal concept of (G, M, I). A *generality* of the intent, denoted by $generality(A)$, is defined as

$$generality(A) = \min_{a \in A} \{support(a)\}.$$

■

If the intent consists of general attributes, its generality becomes high. We expect that it is fairly easy to interpret concepts with such intents. For a concept, on the other hand, if several attributes in the intent have lower supports (that is, specific), it gives a low generality value of the intent. It seems difficult to obtain an understandable interpretation of the concept, even if its intent is relatively compact.

From the definition, for any sets of attributes $A, B \subseteq M$, if $A \subseteq B$ (that is, $A'' \subseteq B''$), then we have $generality(A'') \geq generality(B'')$. That is, the function *generality* is *monotone decreasing*.

For a formal concept, in general, its intent is exactly identified by a subset of the intent. Therefore, such a subset can work as a brief equivalent representation of the intent. With the help of the subset, it is expected we can obtain more adequate meaning of the concept easily. When we present the concept to users, it would be better for clearness to provide the subset rather than the original intent. Such a useful subset is defined as a *generator* of the intent.

Definition 3. (Generator of Intent)
Let (X, A) be a formal concept. For a set of attributes $B \subseteq A$, if $B'' = A$, then B is called a *generator* of A. ■

It is noted here that we have various generators of A. Since a generator can be viewed as one of the concept definitions, such a generator would be desired to be compact and understandable with general attributes for the same reason above. However, we are particularly interested in rare formal concepts with small extents which tend to have larger intents and also larger generators. In order to extract interesting rare concepts with understandable compact generators, we impose a restriction on our generators.

As has been mentioned just above, a generator can work as a concept definition. For example, for a formal concept (X, A) of (G, M, I), let us assume $B = \{b_1, \ldots, b_k\}$ be a generator of A, that is, $B \subseteq A$ and $B'' = A$. The extent X is uniquely identified by the attributes in the generator. Here we pay our attention to the identification process of X by assuming the attributes in B step-by-step. Suppose the attributes in B are linearly ordered as $b_1 \prec \cdots \prec b_k$. The first attribute b_1 identifies the set of objects $\{b_1\}'$. By assuming b_2 with b_1, then, $\{b_1\}'$ is reduced to $\{b_1, b_2\}'$. Adding the next attribute iteratively, we finally obtain the following sequence of object sets:

$$\{b_1\}' \supseteq \{b_1, b_2\}' \supseteq \cdots \supseteq, \{b_1, b_2 \ldots, b_{k-1}\}' \supseteq \{b_1, b_2 \ldots, b_k\}' = X$$

Thus, the initial set of objects, $\{b_1\}'$, is gradually reduced to X, as each attribute is added. The authors claim here that the behavior of this reduction process affects interpretability (or understandability) of the concept definition. More concretely speaking, for a consecutive sets of objects in the sequence, $\{b_1, \ldots, b_{i-1}\}'$ and $\{b_1, \ldots, b_{i-1}, b_i\}'$, if they are almost the same, the attribute b_i has no significant role in characterizing the concept. Therefore, we consider such an attribute b_i *redundant* in the concept definition. Redundant attributes are undesirable because it makes the meaning of the concept unclear. In other words, if the generator B provides a clear definition of the concept, some certain degree of reduction should be observed for each consecutive pair in the above sequence of object sets. This requirement on generators is formalized as follows.

Definition 4. (δ-Generator of Intent)
Let (X, A) be a formal concept of (G, M, I) and $B = \{b_1, \ldots, b_k\}$ a generator of A. Given a *minimum reduction threshold* δ $(0 \leq \delta < 1)$, for any $b_i \in B$ such that $1 \leq i < k$, if

$$\frac{support(\{b_1, \ldots, b_{i+1}\})}{support(\{b_1, \ldots, b_i\})} \leq 1 - \delta,$$

then B is called a δ-*generator* of A. ∎

For a δ-generator B, by assuming each attribute of B in the underlying order, the set of objects concerned with the attributes previously assumed is reduced by *at least* $(100 \times \delta)$ %. Therefore, the parameter δ directly affects the size of generator to be obtained. If a higher δ is given, δ-generators become more compact. It should be noted here that under some parameter settings of δ, we have no δ-generator for a concept.

In the definition of δ-generators, the ordering of attributes in the generator is also important. Let us assume any attribute set to be ordered. If a generator

consists of n attributes, there exist $n!$ possible orderings on them. For example, however, for attributes a_1, a_2 and a_3, even if $\{a_1, a_2, a_3\}$ is a δ-generator, $\{a_1, a_3, a_2\}$ might not. Thus, in order to obtain our δ-generator, we need to extract an adequate ordering among the possibilities. Needless to say, it is quite impractical to examine all of them. According to our intuition, therefore, we impose a constraint on the ordering to be considered. More precisely, the authors suppose that the attributes of a generator we can clearly interpret the meaning should be arranged in *general-to-specific order*. That is, we only accept a δ-generator $\{b_1, \ldots, b_k\}$ such that

$$support(b_1) \geq \cdots \geq support(b_k).$$

In what follows, our δ-generators are assumed to meet the requirement.

Based on the above discussion, we can now summarize our problem of extracting *Top-N Concise Rare Concepts*.

Definition 5. (Top-N Concise Rare Concept Problem)
Let (G, M, I) be a formal context, δ a minimum reduction threshold and ϵ a rareness threshold. Then, the problem of *Top-N rare formal concepts* is to find the set of maximal formal concepts (X, A) satisfying the following two conditions:

Constraints :
(X, A) is ϵ-rare, that is, $|X| = support(A) \leq \epsilon$ and has a δ-generator of A.
Preference :
The value of $generality(A)$ is in the top N among such ones. ■

We discuss below our algorithm for finding Top-N *CRC*s.

3 Finding Top-N Concise Rare Concepts with Depth-First Branch-and-Bound Search

In this section, we present an algorithm for finding Top-N concise rare concepts. It is based on a depth-first branch-and-bound search inspired by a family of efficient *maximum clique algorithms* [18,19,20] and can be viewed as a modified version of those presented in [23,24,25].

3.1 Basic Search Strategy

Let (G, M, I) be a formal context. For each formal concept (X, A) of the context, there always exists a set of attributes $B \subseteq M$ such that $B' = X$ and $B'' = A$. Therefore, by applying the derivation operator to each subset of M, we can *completely* enumerate all formal concepts of (G, M, I).

Let us consider a *linear ordering* $\prec$ on $M = \{m_1, \ldots, m_{|M|}\}$, simply defined as $m_i \prec m_j$ iff $i < j$. It is assumed that for each subset $B \subseteq M$, the elements in B is ordered based on $\prec$.

For a subset of M, $B_i = \{b_{i_1}, \ldots, b_{i_k}\}$, the last element b_{i_k} is denoted by $tail(B_i)$. Furthermore, the set of first ℓ elements, $\{b_{i_1}, \ldots, b_{i_\ell}\}$, is called the ℓ-*prefix* of B_i and is referred to as $prefix(B_i, \ell)$, where $0 \leq \ell \leq k$ and $prefix(B_i, 0)$ is defined as ϕ.

We introduce here a *partial ordering* on 2^M, $\prec_s$, as follows.

Definition 6. (Partial Ordering on 2^M)
Let B_i and B_j be subsets of M. Then B_i *precedes* B_j, denoted by $B_i \prec_s B_j$, iff B_i is a $|B_i|$-prefix of B_j, that is, $B_i = prefix(B_j, |B_i|)$. If B_j is an immediate successor of B_i, then B_j is called a *child* of B_i. ■

It can be easily observed that the partially ordered set $(2^M, \prec_s)$ forms a *tree* with the root node ϕ, called a *set enumeration tree*. For example, given a set of attributes $M = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$, we have a set enumeration tree shown in Figure 1.

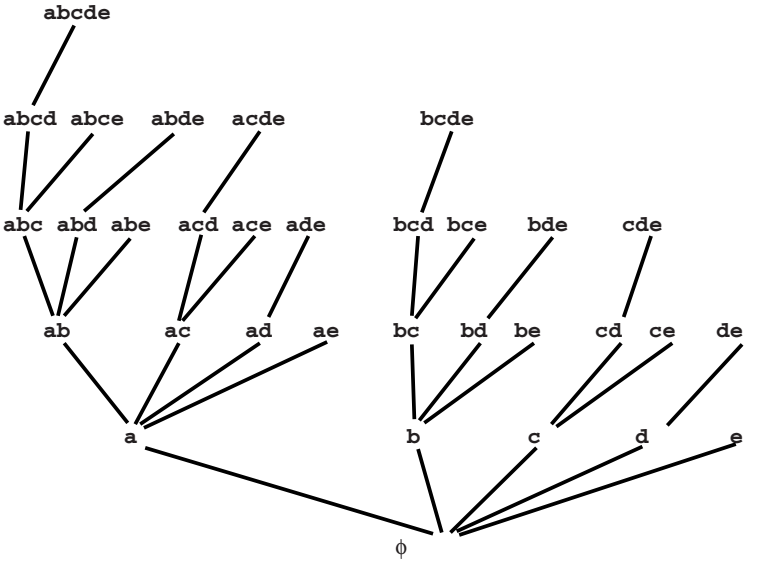


Fig. 1. Set Enumeration Tree

For each (non-leaf) subset $B \subseteq M$ in the tree, its child is simply obtained as $B \cup \{b\}$, where $tail(B) \prec b$. Thus, every subset of M can be generated *systematically without any duplications*, starting with the empty set. More concretely speaking, all of the subsets can be explored in *depth-first* or *breadth-first* manner. We adopt the former in our search, since we can enjoy some effective pruning techniques as will be discussed later.

With the help of the set enumeration tree, we can easily obtain all formal concepts. For each subset $B \subseteq M$, we compute the extent B' and the intent B'' to obtain a formal concept (B', B''), where B can work as a generator of the

intent. In particularly, by arranging the attributes of M in descending order of their support values, that is, general-to-specific order, we can examine all of the possible generators meeting our requirement discussed in the previous section.

Our targets are maximal rare concepts with Top-N concise intents. Therefore, once we find a subset $B \subseteq M$ can produce a rare concept in our depth-first search, any C such that $B \prec_s C$ does not need to be explored. We can immediately backtrack without examining any children of B.

According to the above basic strategy, our search for finding rare formal concepts with Top-N concise intents can be summarized as follows. We examine each (ordered) generator $B \subseteq M$ in a depth-first manner based on the ordering $\prec_s$, starting with the initial subset ϕ. During the search, we maintain a list which stores rare formal concepts with Top-N concise intents already found. That is, the list keeps *tentative* Top-N concepts. For a generator $B \subseteq M$, we first check whether B is a δ-generator or not. If yes, we compute its corresponding formal concept (B', B''), and then check its rareness. If it is rare, then the tentative Top-N list is adequately updated for the concept. That is, the intent B'' has a generality value better than or equal to that of the N-th concept in the tentative list, (B', B'') is newly registered to the list and then backtrack. Otherwise, (B', B'') is just discarded as out of our targets and then we backtrack. If the concept is not rare, a child of B is generated and the same procedure is recursively performed for the child. The procedure is iterated in depth-first manner until no generator remains to be examined.

3.2 Pruning Useless Search Branches

In order to efficiently compute our rare concepts with Top-N concise intents, we must avoid to examine useless generators (sets of attributes) which can never produce our targets. We present here some pruning techniques by which unnecessary search branches are not expanded.

Excluding Non-δ-Generators:

Basically speaking, for a generator $B \subseteq M$, a child of B can be obtained as $B \cup \{b\}$, where $b \in M$ and $tail(B) \prec b$. We call such a b an *extensible candidate* for B. The set of extensible candidates for B is referred to as $cand(B)$, that is, $cand(B) = \{b \in M \mid tail(B) \prec b\}$.

Since each attribute in $cand(B)$ expands a search branch from B, hopeless candidates should be removed from $cand(B)$. Recall here that we are especially interested in only δ-generators. If a generator B is not a δ-generator, any generator C such that $B \prec_s C$ is also not a δ-generator. In that case, we do not have to examine any children of B in our depth-first search. For an attribute $b \in cand(B)$, therefore, if

$$\frac{sup(B \cup \{b\})}{sup(B)} > 1 - \delta,$$

then we can remove b from $cand(B)$ as an useless attribute. After removing such useless b's from $cand(B)$, we actually use the remaining attributes to expand search branches from B.

Avoiding Generation of Duplicate Formal Concepts:

A formal concept of (G, M, I) can be obtained from various sets of attributes in M (i. e. generators). Needless to say, generating duplicate concepts should be avoided for efficient computation. In order to realize it, we need to exclude redundant generators which produce identical concepts and do not need to be examined. We can observe the following simple property of formal concepts based on which we can identify such redundant generators.

Observation 1
Let B be a subset of M. For any attribute $b \in B''\backslash B$, $(B \cup \{b\})'' = B''$ holds. That is, the corresponding formal concepts of $B \cup \{b\}$ and B are identical. ■

Proof:
Let b be an attribute such that $b \in B''\backslash B$. From a property of the derivation operator, $(B \cup \{b\})' = B' \cap \{b\}'$. Moreover, since $b \in B''$, $B' \subseteq \{b\}'$. Therefore, we have $(B \cup \{b\})' = B'$ and then $(B \cup \{b\})'' = B''$. ■

From this observation, when we expand B with an attribute $b \in B''\backslash B$, we have an identical concept (B', B''). This means that $B \cup \{b\}$ can never be a δ-generator of B'' unless we assume $\delta = 0$. However, since the parameter setting of $\delta = 0$ is quite nonsense, it is clear that any attribute $b \in B''\backslash B$ should be removed from $cand(B)$.

Branch-and-Bound Pruning Based on Tentative Top-N Concepts:

Since our target concepts must own the intents with Top-N values of generality, we can exclude any generator which has no possibility of producing those Top-N intents. Such useless generators can be identified based on the tentative Top-N concepts already found in our search.

Let α be the N-th generality value of the Top-N intents already (tentatively) found. That is, the actual (final) N-th value is *at least* α. Assume we are now examining a generator B.

If $generality(B'') < \alpha$, then for any child C of B, $generality(C'') < \alpha$, because the function *generality* is monotone decreasing. Since the children can never produce Top-N intents, there is no need to expand B and we can immediately backtrack.

3.3 Pseudo-code of Algorithm

Based on the above idea, we can design a depth-first branch-and-bound algorithm for finding Top-N concise rare formal concepts. A pseudo-code of our algorithm is presented in Figure 2.

Input :
(G, M, I) : a formal context where
γ : a minimum reduction threshold
ζ : a rareness threshold
N : an integer for Top-N
Output :
$\mathcal{CRC}$: the set of Top-N maximal concise rare concepts

---

procedure main() :
$\mathcal{CRC} \leftarrow \phi$;
$current\_min = 0$;
Arrange the attibutes of M in support descending order ;
TopNCRCFind(ϕ, M, $\mathcal{CRC}$, 0) ;
return $\mathcal{CRC}$;

---

procedure TopNCRCFind(B, $Cand$, $\mathcal{CRC}$) :
$Branch \leftarrow Cand \backslash \{b \in Cand \mid \frac{support(B \cup \{b\})}{support(B)} > 1 - \gamma\}$;
for each $b \in Branch$ such that $tail(B) \prec b$ in predefined order **do**
begin
if $\mathcal{CRC}$ tentatively contains N-th ones and $conciseness(B'') < current\_min$ **then**
break;
endif
$CRC \leftarrow (B, (B \cup \{b\})', (B \cup \{b\})'')$;
if $support((B \cup \{b\})') \leq \zeta$ **then**
TopNListUpdate($\mathcal{CRC}$, CRC) ;
else
TopNCRCFind($B \cup \{b\}$, $Cand \backslash \{b\}$, $\mathcal{CRC}$) ;
endif
end

---

procedure TopNListUpdate($\mathcal{CRC}$, CRC) :
$\mathcal{CRC} \leftarrow \mathcal{CRC} \cup \{CRC\}$;
if $\mathcal{CRC}$ tentatively contains N-th ones **then**
$current\_min \leftarrow N$-th intent value ;
Remove M-th ones from $\mathcal{CRC}$ such that $N < M$;
endif

Fig. 2. Algorithm for Finding Top-N Concise Rare Concepts

4 Preliminary Experimental Results

In this section, we present our preliminary experimental results. Our system has been implemented in C and run on a PC with Intel Core2 Duo E9300 (1.2GHz) and 1GB main memory.

4.1 Dataset

In our experimentation, we have tried to extract Top-N concise rare concepts from a dataset called *BankSearch*.

The dataset *BankSearch* has been released as a benchmark for web document clustering [14]. It consists of web documents (HTML sources) in 11 categories, "*Commercial Banks*", "*Building Societies*", "*Insurance Agencies*", "*Java*", "*C/C++*", "*Visual Basic*", "*Astronomy*", "*Biology*", "*Soccer*", "*Motor Sport*" and "*Sport*". Since each category includes 1,000 documents, the total number of documents in the dataset is 11,000.

As a preprocess, we have first converted each HTML source into a plane text by removing HTML tags. From the text documents, adjectives and adverbs provided in `WordNet` [22] have been eliminated. Furthermore, we have removed a set of stopwords as well. After *Stemming Process* with Porter stemmer [21], we have obtained 1,219 terms as attributes by removing too frequent and too infrequent ones. It should be noted here that the category information is never treated as attributes explicitly.

4.2 Extracted Concise Rare Concepts

We present here some concise rare concepts we have actually extracted.

Under the parameter settings, $\delta = 0.8$ and $\epsilon = 30$, we have tried to find Top-10 *CRC*s. One of the extracted concepts is as follows:

Extent:

$\{2575, 2669, 7062, 7065, 7073, 10009\}$

Intent:

$\{\underline{page}, e, show, develop, manage, \underline{team}, busy,$
$\underline{account}, \underline{insurance}, \underline{article}, issue, board\}$

δ-Generator:

$\{page, team, account, insurance, article\}$

The extent is represented as a set of document IDs. The documents 2575 and 2669 belong to the category of "*Insurance Agencies*". 7062, 7065 and 7073 are in "*Biology*" and 10009 in "*Motor Sport*". That is, this is an instance of crossover concepts.

The concept is evaluated as the 2nd place and the evaluation value of the intent is 1373. The extent consists of just 6 documents and the intents 12 attributes (terms), where 5 of them are the elements of a δ-generator. The attributes in the intent are listed in general-to-specific order.

The δ-generator tells us that these documents are mainly characterized by the 5 terms. In fact, all of the documents are concerned with topics in which "*insurance*" is important. Thus, the generator would be helpful for understanding the concept.

As another example, we have had a *CRC* extracted under the parameter settings, $\delta = 0.5$ and $\epsilon = 20$.

Extent:

$\{230, 620, 748, 3020, 3148, 3261, 4105, 7065, 7267, 9011\}$

Intent:

$\{\underline{page}, site, \underline{make}, \underline{02}, \underline{contact}, \underline{UK}, \underline{e}, \underline{2001}, \underline{application}, \underline{update}, access, support\}$

δ-Generator:

$\{page, make, 02, contact, UK, e, 2001, application, update\}$

Table 1. Computation Time for *BankSearch*

$\epsilon = 10$

δ	Computation Time (sec.)
0.5	0.75
0.6	0.53
0.7	0.51
0.8	0.43
0.9	0.43

$\epsilon = 20$

δ	Computation Time (sec.)
0.5	0.64
0.6	1.05
0.7	0.49
0.8	0.43
0.9	0.40

$\epsilon = 30$

δ	Computation Time (sec.)
0.5	0.57
0.6	0.51
0.7	0.47
0.8	0.49
0.9	0.39

$\epsilon = 40$

δ	Computation Time (sec.)
0.5	0.64
0.6	0.47
0.7	0.47
0.8	0.47
0.9	0.40

$\epsilon = 50$

δ	Computation Time (sec.)
0.5	0.63
0.6	0.49
0.7	0.49
0.8	0.51
0.9	0.45

It should be noted here that we have not obtained a similar generator for another year, e.g. 2000. This implies that something about *contact*, UK, *application* and *update* happens particularly in 2001. Using this valuable information as a trigger, we might be able to analyze this concept more deeply and precisely.

In addition to those concepts above, we have obtained an interesting *CRC*. Its extent consists of just 5 documents in the categories, "*Astronomy*", "*Biology*" and "*Motor Sport*". The intent contains 108 attributes. Needless to say, it is quite difficult to obtain a clear meaning of the concept only from those many attributes. However, the concept can be characterized by a very compact generator with just 6 attributes, as our system has outputted. The generator would be a great help in clearly understanding the concept.

4.3 Computational Performance

Computational performance of our algorithm is summarized in Table 1.

Our algorithm took less that 1 second for all problems except just one ($\epsilon = 20$ and $\delta = 0.6$). Thus, our restricted formation process of concepts can make the computation very efficient, still keeping the clearness of concepts.

As is well known, the intent of a formal concept is an equivalent notion of *closed itemset* in the field of Data Mining [17]. With the help of closed itemset miners, e.g. `LCM` [7] and `D-Miner` [15], therefore, we can obtain formal concepts. Such systems are, however, not helpful for finding our rare concepts. Given a parameter *minsup*, they can usually enumerate all *frequent* closed itemsets in the sense that their supports are greater than or equal to *minsup*. If we try to find our rare concepts with those miners, we have to give a quite small value of *minsup*. As a well known fact, under such a low *minsup*, frequent closed itemset miners are quite useless because the number of frequent closures is enormous. In fact, we have verified that these famous excellent miners cannot enumerate frequent closures in reasonable time (within 6 hours) even under a min_sup value larger than our rareness threshold. This is a remarkable advantage of our method.

5 Concluding Remarks

We discussed in this paper a problem of finding Top-N concise rare concepts and designed a depth-first branch-and-bound algorithm for the problem. Although it is well known that rare concepts provide us valuable information, they in general tend to have longer intents which make meanings of the concepts unclear. In order to solve this issue, we imposed a restriction on our formation process of concepts so that we can obtain more understandable concepts. Our preliminary experimental results showed that extracted concepts are conceptually clear in the sense that they can be characterized by δ-generators with general attributes. Furthermore, our restricted formation brought us efficient computation of concepts, still keeping their interpretability.

In our experimentation, we verified that crossover concepts can be detected by our algorithm. However, the current formalization is not only for such crossover concepts. Such a tailored algorithm is currently under investigation.

Furthermore, one might claim that our formation process of concepts is too restrictive. It is actually true that the pre-defined ordering on attributes strongly affects which kind of concepts we can detect. In order to make our method more useful, we have to further investigate the ordering from various viewpoints as well as generality of attributes.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis - Mathematical Foundations, 284 p. Springer, Heidelberg (1999)
2. Ganter, B., Stumme, G., Wille, R. (eds.): Formal Concept Analysis – Foundations and Applications. LNCS (LNAI), vol. 3626, 348 p. Springer, Heidelberg (2005)
3. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, pp. 207–216 (1993)
4. Lakhal, L., Stumme, G.: Efficient Mining of Association Rules Based on Formal Concept Analysis. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 180–195. Springer, Heidelberg (2005)
5. Wang, J., Han, J., Pei, J.: CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In: Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining - KDD 2003, pp. 236–245 (2003)
6. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent Pattern Mining - Current Status and Future Directions. Data Mining and Knowledge Discovery 15(1), 55–86 (2007)
7. Uno, T., Kiyomi, M., Arimura, H.: LCM ver. 2: Efficient Mining Algorithm for Frequent/Closed/Maximal Itemsets. In: Proc. of IEEE ICDM 2004 Workshop - FIMI 2004 (2004), http://sunsite.informatik.rwth-aachen.de/verb+Publications/CEUR-WS//Vol-126/
8. Dong, G., Li, J.: Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In: Proc. of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining - KDD 1999, pp. 43–52 (1999)
9. Alhammady, H., Ramamohanarao, K.: Using Emerging Patterns and Decision Trees in Rare-Class Classification. In: Proc. of the 4th IEEE Int'l Conf. on Data Mining - ICDM 2004, pp. 315–318 (2004)
10. Bay, S.D., Pazzani, M.J.: Detecting Group Differences: Mining Contrast Sets. Data Mining and Knowledge Discovery 5(3), 213–246 (2001)
11. Novak, P.K., Lavrac, N.: Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining. The Journal of Machine Learning Research Archive 10, 377–403 (2009)
12. Li, A., Haraguchi, M., Okubo, Y.: Implicit Groups of Web Pages as Constrained Top-N Concepts. In: Proc. of the 2008 IEEE/WIC/ACM Int'l Conf. on Web Intelligence and Intelligent Agent Technology Workshops, pp. 190–194 (2008)
13. Nebel, B.: Reasoning and Revision in Hybrid Representation. Springer, Heidelberg (1989)
14. Sinka, M.P., Corne, D.W.: A Large Benchmark Dataset for Web Document Clustering. In: Soft Computing Systems: Design, Management and Applications. Series of Frontiers in Artificial Intelligence and Applications, vol. 87, pp. 881–890 (2002)
15. Besson, J., Robardet, C., Boulicaut, J.: Constraint-Based Concept Mining and Its Application to Microarray Data Analysis. Intelligent Data Analysis 9(1), 59–82 (2005)

16. Szathmary, L., Napoli, A., Valtchev, P.: Towards Rare Itemset Mining. In: Proc. of the 19th IEEE Int'l Conf. on Tools with Artificial Intelligence - ICTAI 2007, pp. 305–312 (2007)
17. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Information Systems 24(1), 25–46 (1999)
18. Tomita, E., Kameda, T.: An Efficient Branch-and-Bound Algorithm for Finding a Maximum Clique with Computational Experiments. Journal of Global Optimization 37, 95–111 (2007)
19. Tomita, E., Seki, T.: An Efficient Branch and Bound Algorithm for Finding a Maximum Clique. In: Calude, C.S., Dinneen, M.J., Vajnovszki, V. (eds.) DMTCS 2003. LNCS, vol. 2731, pp. 278–289. Springer, Heidelberg (2003)
20. Fahle, T.: Simple and Fast: Improving a Branch-and-Bound Algorithm for Maximum Clique. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 485–498. Springer, Heidelberg (2002)
21. Porter, M.F.: An Algorithm for Suffix Stripping. Program 14(3), 130–137 (1980)
22. Fellbaum, C. (ed.): WordNet - An Electronic Lexical Database. MIT Press, Cambridge (1998)
23. Haraguchi, M., Okubo, Y.: An Extended Branch-and-Bound Search Algorithm for Finding Top-N Formal Concepts of Documents. In: Washio, T., Satoh, K., Takeda, H., Inokuchi, A. (eds.) JSAI 2006. LNCS (LNAI), vol. 4384, pp. 276–288. Springer, Heidelberg (2007)
24. Haraguchi, M., Okubo, Y.: A Method for Pinpoint Clustering of Web Pages with Pseudo-Clique Search. In: Jantke, K.P., Lunzer, A., Spyratos, N., Tanaka, Y. (eds.) Federation over the Web. LNCS (LNAI), vol. 3847, pp. 59–78. Springer, Heidelberg (2006)
25. Okubo, Y., Haraguchi, M.: Finding Conceptual Document Clusters with Improved Top-N Formal Concept Search. In: Proc. of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence - WI 2006, pp. 347–351 (2006)

Conditional Functional Dependencies: An FCA Point of View

Raoul Medina and Lhouari Nourine

Université Blaise Pascal – LIMOS
Campus des Cézeaux,
63173 Aubière Cedex, France
{medina,nourine}@isima.fr

Abstract. Conditional Functional Dependencies (CFDs) are Functional Dependencies (FDs) that hold on a fragment relation of the original relation. In [17], the hierarchy between CFDs, association rules and some other dependencies have been shown.

This paper exhibits the relation between CFDs and FCA. Given a many-valued relation we define a labeled lattice which gives a synthetic representation of the hierarchy of dependencies. Moreover, a formal concept in the nominal scaling of the relation is an instance of a closed set in the labeled lattice. Pure CFDs correspond to edges in this labeled lattice. We exhibit a monotone function on CFDs allowing search and pruning strategies. We also show that transitive edges induce redundant CFDs.

1 Introduction

Dependency theory is one of the major subjects of database theory and has been traditionally used to optimize queries, prevent invalid updates and to normalize databases. Originally, dependency theory has been developed for uninterpreted data and served mainly database conception purposes [10]. The Functional Dependencies introduced by Codd were generalized to Equality Generating Dependencies (EGD) [4]. In [3], dependencies over interpreted data (i.e. equality requirements are replaced by constraints of an interpreted domain) were introduced and generalized the EGDs into Constraint-Generating-Dependencies (CGD). In [7], the authors present a particular case of CGDs: Conditional Functional Dependencies (CFDs), for data cleaning purposes. CFDs are dependencies which hold on instances of the relations. Constraint used in CFDs is the equality and allows to fix particular constant values for attributes. Basically, CFDs can be viewed as FDs which hold on a fragment relation of the original instance relation, this fragment relation being characterized by the constraints to be applied on the attributes. Those constraints represent a selection operation on the relation. All these works focused mainly on implication analysis and axiomatizability.

Discovery of dependencies existing in an instance of a relation received considerable interest as it allowed automatic database analysis. Knowledge discovery and data mining [1], database management [6],[9], reverse engineering [19] and

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 161–176, 2010.

query optimization [20] are among the main applications benefiting from efficient dependencies discovery algorithms. New dependencies on instances of relations were defined. We cite (among others): Association Rules (AR) [1] which are dependencies holding on particular values of attributes, and Approximate Functional Dependencies (AFD) [16] which are FDs which almost hold on a given relation. Note that AFDs have also applications in database design [11]. For those latter dependencies, several measures of approximateness were defined, expressing the interest of the dependency.

Recently, we have shown in [17] that there exists a hierarchy among those dependencies: FDs and CFDs are the union of ARs, AFD are the union of approximate AR. As a consequence, CFDs can be used as a *synthetic representation* of ARs.

In this paper, we present CFDs from a lattice point of view. We define a labeled lattice in which CFDs correspond to cover edges of the lattice. We present some properties of this lattice and show the link between this lattice and the traditional concept lattice of a relation. We show that a concept (in the FCA sense [14]) is an *instance* of a closed set in the labeled lattice. As a consequence, we show that the lattice of CFDs is a synthetic representation of the concept lattice.

The paper is organized as follows: in section 2 we recall definitions and results on CFDs. Section 3 presents the labeled lattice of CFDs and some properties of this lattice. Section 4 presents the link between the lattice of CFDs and traditional concept lattice of a relation. Section 5 discusses the results of our paper.

2 Background, Definitions and Preliminary Results

2.1 Definitions

Let R be a relation schema defined over a set of attributes $Attr(R)$. The domain of each attribute $A \in Attr(R)$ is denoted by $Dom(A)$. For an instance r of R, a

r	A	B	C	D	E	F
t_1	a_0	b_0	c_0	d_0	e_0	f_0
t_2	a_1	b_1	c_1	d_0	e_1	f_1
t_3	a_1	b_1	c_2	d_1	e_2	f_2
t_4	a_2	b_1	c_2	d_1	e_2	f_3
t_5	a_2	b_1	c_2	d_2	e_3	f_4
t_6	a_2	b_2	c_3	d_3	e_4	f_5
t_7	a_3	b_3	c_4	d_3	e_4	f_5
t_8	a_4	b_0	c_5	d_4	e_5	f_6
t_9	a_5	b_2	c_6	d_5	e_6	f_7
t_{10}	a_6	b_3	c_7	d_6	e_7	f_8

Fig. 1. An instance relation r of the schema R

tuple $t \in r$ and X a sequence of attributes, we use $t[X]$ to denote the projection of t onto X.

An *FD* $X \rightarrow Y$, where $X, Y \subseteq Attr(r)$, is satisfied by r, denoted by $r \vDash X \rightarrow Y$, if for all pairs of tuples $t_1, t_2 \in r$ we have: if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$. In other words, all pairs of tuples *agreeing* on attributes X will agree on attributes Y.

A *CFD* φ defined on R is a pair $(X \rightarrow Y, T_p)$, where (1) $X \rightarrow Y$ is a standard FD, referred to as the FD embedded in φ; and (2) T_p is a tableau with attributes in R, referred to as the pattern tableau of φ, where for each $A \in Attr(R)$ and each pattern tuple $t_p \in T_p$, $t_p[A]$ is either:

- a constant 'a' in $Dom(A)$,
- an unnamed variable $\top$ that draws values from $Dom(A)$,
- or an empty variable $\perp$ which indicates that the attribute does not contribute to the pattern (i.e. $A \notin X \cup Y$). [1]

For any constant a of an attribute we have: $\perp \leq a \leq \top$. We define the *pattern intersection operator* $\sqcap$ of two tuples as:

$$t_1 \sqcap t_2 = t_p \text{ such that } \forall A \in Attr(r), \begin{cases} t_p[A] = t_1[A], & \text{if } t_1[A] \leq t_2[A] \\ t_p[A] = t_2[A], & \text{if } t_1[A] > t_2[A] \\ t_p[A] = \perp, & \text{otherwise.} \end{cases}$$

We define the *pattern restriction to attributes* X of a tuple t, denoted by $t(X)$ as:

$$t(X) = t_p \text{ such that } \forall A \in Attr(r), \begin{cases} t_p[A] = t[A], & \text{if } A \in X \\ t_p[A] = \perp, & \text{otherwise.} \end{cases}$$

The pattern restriction to attributes X for a pattern tableau, denoted by $\Pi_X(T_p)$ is:

$$\Pi_X(T_p) = \{t_p(X) \mid t_p \in T_p\}$$

We define a subsumption operator $\sqsubseteq$ over pattern tuples t_1 and t_2:

$$t_1 \sqsubseteq t_2 \text{ if and only if } \forall A \in Attr(R), t_1[A] \leq t_2[A].$$

In other words, $t_1 \sqsubseteq t_2$ if and only if $t_1 \sqcap t_2 = t_1$. We define the special tuple Top as the tuple with value $\top$ on all attributes, i.e. $t_p \sqsubseteq Top$ for any pattern tuple t_p.

An instance r of R satisfies the CFD φ, denoted by $r \vDash \varphi$, if for each tuple t_p in the pattern tableau T_p of φ, and for each pair of tuples t_1, t_2 in r, if $t_1(X) = t_2(X) \sqsupseteq t_p(X)$, then $t_1(Y) = t_2(Y) \sqsupseteq t_p(Y)$. In other words, a CFD is an FD satisfied by a fragment relation.

A pattern tuple t_p defines a fragment relation of r:

$$r_{t_p} = \{t \in r \mid t_p \sqsubseteq t\}.$$

[1] In [12], empty variables are not present in the pattern tableau.

We will denote by r_{T_p} the fragment relation containing all tuples of r satisfying at least one of the patterns in T_p. Note that given a CFD $(X \rightarrow Y, T_p)$, we thus have $r_{T_p} \models X \rightarrow Y$ and $r - r_{T_p} \not\models X \rightarrow Y$.

A pattern tableau can thus be seen as a selection query on a relation returning a fragment of the relation. Two pattern tableaux will be said *equivalent* if and only if they return the same fragment relation.

Lemma 1 (see [17]). *For any pattern tableau T_P there exists an equivalent pattern tableau $T_{P_{Const}}$ such that pattern tuples in $T_{P_{Const}}$ contain either constant or empty variables.*

Consequence of previous lemma is that, without loss of generality, we will focus on CFDs which tableaux contain only constant or empty attributes. In [17], we show how to generate all equivalent tableaux from such constant tableaux.

Example 1. In the relation r of Figure 1, $A \rightarrow B$ is a CFD which holds on the fragment relation $\{t_1, t_2, t_3, t_7, t_8, t_9, t_{10}\}$. Its constant pattern tableau is:
$\{ (a_0,\perp,\perp,\perp,\perp,\perp), (a_1,\perp,\perp,\perp,\perp,\perp), (a_3,\perp,\perp,\perp,\perp,\perp),$
$(a_4,\perp,\perp,\perp,\perp,\perp), (a_5,\perp,\perp,\perp,\perp,\perp), (a_6,\perp,\perp,\perp,\perp,\perp)\ \}$

2.2 X-Complete Relations

Using the intersection operator over tuples we could build the tuples lattice of a relation. A closed tuple will thus subsume all tuples agreeing on the same values, i.e. the values of non empty variables in the closed tuple. This notion of set of tuples agreeing on the same values for a given set of attributes X has already been defined in database theory for horizontal decomposition purposes [11] or for FDs discovery [16]. We thus use their definition to define the different closure operators we use in this paper.

Definition 1 (X-complete property [11]). *The relation r is said to be* X-complete *if and only if $\forall\, t_1, t_2 \in r$ we have $t_1[X] = t_2[X]$.*

In other words, a relation is X-complete if all tuples agree on the attributes X. Note that they might also agree on other attributes: this constitutes the pattern of r.

In [17], several definitions based on X-complete relations were used. We recall them briefly in the following table.

Definition	Notation	Formal definition
X-Complete pattern	$\gamma(X, r)$	$\sqcap \{t \in r\}$ with r is X-complete
X-complete horizontal decomposition	$R_X(r)$	$\{r' \subseteq r \mid r'$ is X-complete$\}$
Set of X-patterns	$\Gamma(X, r)$	$\{\gamma(X, r') \mid r' \in R_X(r)\}$.
Closure operator	$\theta(X, r)$	$\{A \in Attr(r) \mid \forall t_p \in \Gamma(X, r), t_p[A] \neq \perp\}$

Definition 2 (X-closed relation). *An X-complete fragment relation $r' \subseteq r$ is said* X-closed *if $\theta(X, r') = X$.*

Using the closure operator $\theta(X, r)$, we can trivially characterize FDs[2].

Property 1 (See [17]). Let $A \notin X$. We have $r \vDash X \rightarrow A$ (i.e. $X \rightarrow A$ is an FD of r) if and only if $A \in \theta(X, r)$.

Note that the closure operator $\theta(X, r)$ is equivalent to the closure of a set of attributes X using FD of r.

Proposition 1 (see [17]). *Let $r' \subseteq r$. Then r' is X-complete if and only if r' is $\theta(X, r)$-complete.*

A consequence is that X and $\theta(X, r)$ define the same X-complete horizontal decomposition of r. This leads to the following corollary.

Corollary 1 (see [17]). $\Gamma(X, r) = \Gamma(\theta(X, r), r)$.

Example 2. In the relation of figure 1, the fragment relation $\{t_3, t_4\}$ is D-complete. Its pattern is $\gamma(D, \{t_3, t_4\}) =$ $(\bot,b_1,c_2,d_1,e_2,\bot)$. The set of all D-patterns is:
$\{$ $(\bot,\bot,\bot,d_0,\bot,\bot)$, $(\bot,b_1,c_2,d_1,e_2,\bot)$, $(a_2,b_1,c_2,d_2,e_3,f_4)$, $(\bot,\bot,\bot,d_3,e_4,f_5)$,
$(a_4,b_0,c_5,d_4,e_5,f_6)$, $(a_5,b_2,c_6,d_5,e_6,f_7)$, $(a_6,b_3,c_7,d_6,e_7,f_8)\}$.
The closure of D in r is $\theta(D, r) = D$.

2.3 Hierarchy of Dependencies

Here we recall the relation between CFDs and other usual dependencies (see [17] for details).

FDs: An FD $X \rightarrow A$ can be seen as a CFD $(X \rightarrow A, t_p)$ where t_p is a single tuple with no constants, i.e. $t_p = Top(X \cup \{A\})$. In other words, $r \vDash X \rightarrow A$ if and only if $r_{X \rightarrow A} = r$ (and thus $r_{X \not\rightarrow A} = \emptyset$).

ARs: An AR is a dependency $(X_1 = b_1) \wedge \cdots \wedge (X_k = b_k) \rightarrow (A = a)$ meaning that for any tuple $t \in r$, if $t[X_1] = b_1$ and $\cdots$ and $t[X_k] = b_k$ then $t[A] = a$. An AR can be expressed using a CFDs which tableau consists of a single pattern tuple containing only constant or empty values.

Theorem 1 (Hierarchy [17]). *Let r be a relation, $X \subseteq Attr(r)$, $A \in Attr(r) \setminus X$ and $T_p = \{t_p \in \Gamma(X, r) \mid t_p[A] \neq \bot \}$. The following assertions are equivalent:*

1. *$(X \rightarrow A, T_p)$ is a CFD of r*
2. *$X \rightarrow A$ is an FD of r_{T_p}*
3. *For any $r' \in R_X(r_{T_p})$, $(X \rightarrow A, \gamma(X, r'))$ is an AR of r*

Theorem 1 leads to a hierarchy among those dependencies:

- an AR $(X \rightarrow A, t_p)$ is a dependency that holds on at least one fragment relation of r which is X-complete.

[2] Note that this closure operator is the same that the one obtained using the agree sets [5].

- a CFD $(X \rightarrow A, T_p)$ is a dependency that holds on some fragment relations of r which are X-complete. It can thus be viewed as the union of ARs holding on those fragment relations.
- an FD is a dependency that holds on all fragment relations of r which are X-complete. It can thus be viewed as the union of ARs holding on all those fragment relations.

Definition 3 (Y-Valid and Y-Invalid X-complete pattern tuples). *Given r' an X-complete fragment relation of r. Its corresponding X-complete pattern tuple $t_p = \gamma(X, r')$ is said to be* Y-valid towards Y *if and only if attributes Y are defined in t_p.*
We denote by $T_p(X \rightarrow Y)$ the set of all Y-valid X-complete pattern tuples. More formally: $T_p(X \rightarrow Y) = \{t_p \in \Gamma(X, r) \mid \forall A \in Y, t_p[A] \neq \perp\}$.
Dually, we denote by $T_p(X \not\rightarrow Y)$ the set of all Y-invalid X-complete pattern tuples: $T_p(X \not\rightarrow Y) = \Gamma(X, r) - T_p(X \rightarrow Y)$.

In [17], a classification of dependencies was done using A-Valid and A-Invalid X-complete pattern tuples. The following table summarizes this classification.

Dependency	Type	$T_p(X \rightarrow A)$?	$T_p(X \not\rightarrow A)$?
$X \not\rightarrow A$		$T_p(X \rightarrow A) = \emptyset$	
$X \rightarrow A$	Pure AR	$\mid T_p(X \rightarrow A) \mid = 1$	$\mid T_p(X \not\rightarrow A) \mid \geq 1$
	Pure CFD	$\mid T_p(X \rightarrow A) \mid \geq 2$	$\mid T_p(X \not\rightarrow A) \mid \geq 1$
	FD		$T_p(X \not\rightarrow A) = \emptyset$

We recall that a CFD is said to be a pure AR if its pattern tableau contains only a single pattern tuple (i.e this CFD corresponds to a unique AR). A CFD is said to be a pure CFD if it is the union of at least two ARs and does not generalize into an FD (see [17] for details).

3 The X-Complete Partitions Lattice

3.1 Labeled Lattice of Closed Sets

In this section, we show that considering closed sets $\theta(X, r)$ is sufficient to compute all CFDs.

Definition 4 (Generator and minimal generator). *Given $Y \subseteq Attr(r)$, a subset $X \subseteq Y$ is said to be a* generator *of $\Gamma(Y, r)$ if and only $\Gamma(X, r) = \Gamma(Y, r)$. It is said to be a* minimal generator *if and only if there does not exist $Z \subset X$ such that $\Gamma(Z, r) = \Gamma(Y, r)$. By extension, we also say that X is a minimal generator of $\theta(X, r)$.*

Next Theorem states that if we consider CFDs which left-part is a closed set, then the remaining CFDs could be obtained by computing the generators of these closed sets.

Theorem 2. *Let r be a relation, $X, Y \subseteq Attr(r)$ such that $X \cap Y = \emptyset$. We have: $r \vDash (X \rightarrow Y, T_p(X \rightarrow Y))$ if and only if $r \vDash (\theta(X,r) \rightarrow Y, T_p(X \rightarrow Y))$.*

Proof. Follows directly from Corollary 1. □

Generating CFDs of the form $(\theta(X,r) \rightarrow Y, T_p(X \rightarrow Y))$ is thus sufficient. Indeed, all other CFDs can be found as $(Z \rightarrow Y, \Pi_Z(T_p(X \rightarrow Y)))$ where Z is a generator of $\theta(X,r)$. Moreover, if $Z \neq \theta(X,r)$ then $\theta(Z,r) \rightarrow \theta(X,r) - Z$ is an FD. As a consequence, all closed sets of the relation r (defined with the operator $\theta(X,r)$) are potential antecedent of a CFD. We could thus consider the lattice of closed sets of r as the search space for CFDs.

Theorem 3 (Transitivity). *Let $X \subset Y \subset Z$ be subsets of attributes. Then*

$$T_p(X \rightarrow Z) = T_p(X \rightarrow Y) \cap \Pi_X(T_p(Y \rightarrow Z))$$

Proof. Let $t_p \in T_p(X \rightarrow Z)$. We show that r_{t_p} belongs both to $r_{T_p(X \rightarrow Y)}$ and $r_{T_p(Y \rightarrow Z)}$. Since $t_p \in T_p(X \rightarrow Z)$, r_{t_p} is X-complete and Z-complete. And from $X \subset Y \subset Z$ we deduce that r_{t_p} is also Y-complete. Thus $r_{t_p} \subseteq r_{T_p(X \rightarrow Y)}$ and $r_{t_p} \subseteq r_{T_p(Y \rightarrow Z)}$.

Now, consider $t_p \in T_p(X \rightarrow Y) \cap \Pi_X(T_p(Y \rightarrow Z))$. We show that $r_{t_p} \subseteq r_{T_p(X \rightarrow Z)}$. Since $t_p \in T_p(X \rightarrow Y) \cap \Pi_X(T_p(Y \rightarrow Z))$, we have r_{t_p} is X-complete, Y-complete and Z-complete. Thus, $r_{t_p} \subseteq r_{T_p(X \rightarrow Z)}$. □

Example 3. Let us consider the CFD $D \rightarrow B$. The CFD $D \rightarrow B$ is a "transitive" CFD since we have $D \rightarrow E$ and $DE \rightarrow BC$. Thus, the tableau of $D \rightarrow B$ can be computed from the tableaux of $D \rightarrow E$ and $DE \rightarrow BC$ as follows: $T_p(D \rightarrow B) = T_p(D \rightarrow E) \cap \Pi_D(T_p(DE \rightarrow BC))$. We obtain $T_p(D \rightarrow B) = \{$ $(\perp,\perp,\perp,d_1,\perp,\perp)$, $(\perp,\perp,\perp,d_2,\perp,\perp)$, $(\perp,\perp,\perp,d_4,\perp,\perp)$, $(\perp,\perp,\perp,d_5,\perp,\perp)$, $(\perp,\perp,\perp,d_6,\perp,\perp)\}$ (see Figure 2 for the tableaux of $D \rightarrow E$ and $DE \rightarrow BC$).

From Theorems 3 we deduce that "transitive" CFDs can be computed from CFDs of the form $(\theta(X,r) \rightarrow \theta(Y,r) \setminus \theta(X,r), T_p(X \rightarrow Y))$ such that $\theta(Y,r)$ covers $\theta(X,r)$ in the lattice of closed sets of r.

Definition 5 (Tableau of a closed set). *Given a closed set X of r, we denote by $T_p(X)$ the set of pattern tuples of all X-complete partitions which are X-closed. More formally: $T_p(X) = \{t_p \in \Gamma(X,r) \mid \forall A \notin X, t_p[A] = \perp\}$.*

Note that the tableau of a closed set might be empty. In this case, the tableau contains only one tuple $(\perp,\perp,\perp,\perp,\perp,\perp)$.

Example 4. For the relation in figure 1, the tableau of the closed set BC is: $T_p(BC) = \{$ $(\perp,b_1,c_2,\perp,\perp,\perp)\}$. The closed set DE has an empty tableau: $T_p(DE) = \{$ $(\perp,\perp,\perp,\perp,\perp,\perp)\}$.

Definition 6 (X-complete partitions lattice). *Let $\mathcal{L}(r,\theta)$ be the lattice of closed sets of a many-valued relation r wrt. Closure operator θ. The lattice of X-complete partitions is $\mathcal{L}(r,\theta)$ completed with two labeling functions: $T_p(X)$ for a closed set X, $T_p(X \rightarrow Y \setminus X)$ for an edge (X,Y) of the Hasse diagram of $\mathcal{L}(r,\theta)$.*

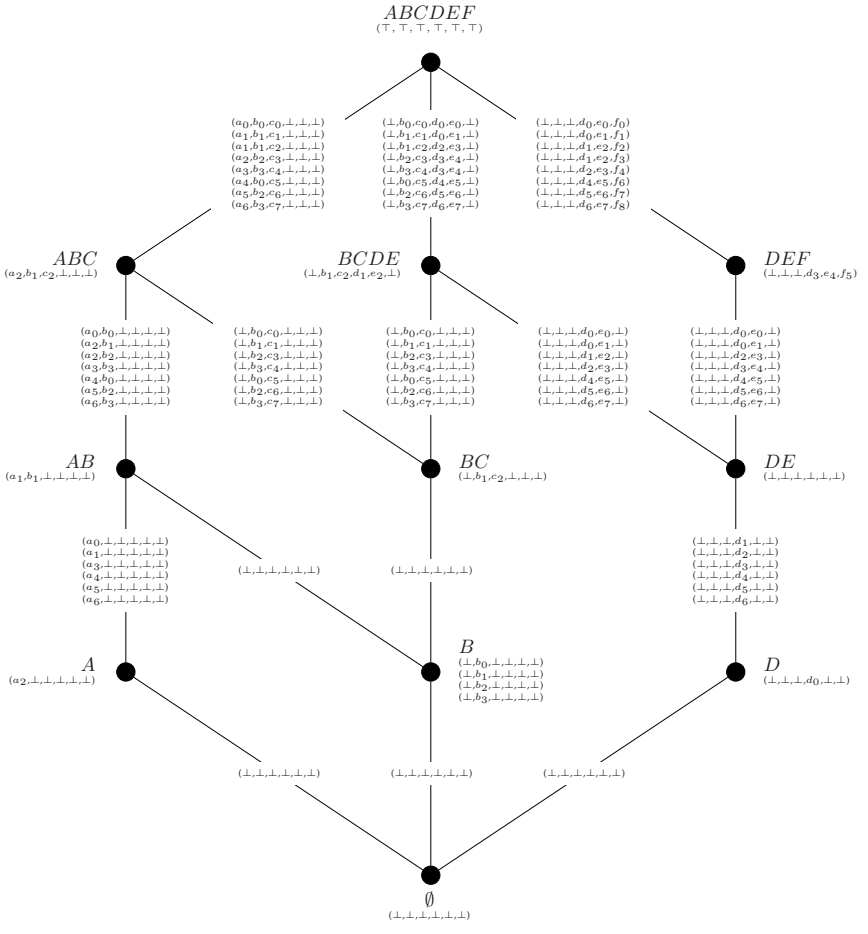


Fig. 2. The X-complete partitions lattice of relation r. Label of a closed set represents the set of formal concepts which are particular instances of this global concept. Label of an edge (X, Y) represents the tableau of the CFD $X \to Y$.

Figure 2 shows the lattice of X-complete partitions for the relation r in figure 1.

Property 2. Let X be a closed set in r. For any $Y \subseteq Attr(R) \setminus X$ we have: $T_p(X) \cap T_p(X \to Y) = \emptyset$.

Previous property states that a pattern tuple present in the tableau of a closed set X cannot be present in the tableau of a CFDs having X as antecedent.

Proposition 2. *Let X be a closed set and $Y_1, \cdots, Y_k$ be all closed sets covering X. Then we have:*

$$r = r_{T_p(X)} \cup \bigcup_{i \in 1..k} r_{T_p(X \to Y_i)}$$

Proof. It is immediate to see that $r \supseteq r_{T_p(X)} \cup \bigcup_{i \in 1..k} r_{T_p(X \to Y_i)}$. Let us prove that $r \subseteq r_{T_p(X)} \cup \bigcup_{i \in 1..k} r_{T_p(X \to Y_i)}$.

Consider $r' \in R_X(r)$. Let us show that $\gamma(X, r) \in T_p(X) \cup \bigcup_{i \in 1..k} T_p(X \to Y_i)$. If $\gamma(X, r') = X$ then it is straightforward that $\gamma(X, r') \in T_p(X)$. Suppose that $\gamma(X, r') \neq X$, then we distinguish two cases:

- $\exists j \in 1..k$ such that $\gamma(X, r') = Y_j$. Thus $\gamma(X, r') \in T_p(X \to Y_j)$.
- $\gamma(X, r') = Z$ with $Z \neq Y_j$ for any $j \in 1..k$. There thus exists $j \in 1..k$ such that $X \subset Y_j \subset Z$. Since $\gamma(X, r') \in T_p(X \to Z)$ and by Theorem 3 we thus have $\gamma(X, r') \in T_p(X \to Y_j)$. □

Consequence is that a closed set X splits the relation in two parts: $r_{T_p(X)}$ where X is *never* antecedent of a CFD or of an AR and the remaining of the relation where X is *always* antecedent of a CFD.

3.2 Simplified Labeled Lattice

In this section we show that some redundancy appears in the labeling of the lattice and we propose a simplified labeling to avoid such redundancy.

Theorem 4 (Monotony). *Let X be a closed set and r a many-valued relation. Then for any $Y \subseteq Attr(R)$ such that $X \subseteq Y$ we have :*

$$\forall A \in Attr(R) \setminus Y, \quad r_{T_p(X \to A)} \subseteq r_{T_p(Y \to A)}$$

Proof. Let t_p be a pattern in $T_p(X \to A)$. We show that r_{t_p} is partitioned into several Y-complete fragment relations that belong to $r_{T_p(Y \to A)}$. Clearly, r_{tp} is $X \cup A$-complete. Thus $\Gamma(Y, r_{tp}) \subseteq T_p(Y \to A)$. □

A CFD is an FD which holds on a fragment relation of the original relation. Theorem 4 states that whenever we add attributes Y in the antecedent part of a CFD $X \to A$, if the new CFD $X \cup Y \to A$ still holds then the fragment relation $r_{T_p(X \cup Y \to A)}$ on which it holds can be partitioned in two: the fragment relation $r_{T_p(X \to A)}$ of the original CFD and a new fragment relation $r_{T_p(X \cup Y \to A)} \setminus r_{T_p(X \to A)}$. Only this latter fragment relation adds non redundant information for the understanding of the *new* CFD $X \cup Y \to A$. The idea of the *simplified labeling* of the X-complete partitions lattice is thus to label edges of the lattice only with this new information. Note that $r_{T_p(X \cup Y \to A)} \setminus r_{T_p(X \to A)} \subseteq r_{T_p(X \not\to A)}$. The simplified label to use is thus the patterns of $T_p(X \cup Y \to A)$ which selects tuples in $r_{T_p(X \not\to A)}$.

Another redundancy appears when a pattern tuple in the tableau of a CFD $X \to Y$ selects a single tuple (i.e. the fragment relation has size 1).

Proposition 3. *Let $(X \to Y, T_p(X \to Y))$ be a CFD of r and $t_p \in T_p(X \to Y)$ such that $r_{t_p} = \{t\}$. Then for any CFD $Z \to W$ such that $X \subseteq Z$, we have $t(Z) \in T_p(Z \to W)$.*

In other words, X is a local key for the fragment relation defined by t_p. The tuple selected by t_p will thus appear in any CFD which antecedent contains X.

Definition 7 (Simplified X-complete partitions lattice). *The simplified X-complete partitions lattice is obtained by removing in the label of edges of the X-complete partition lattice, pattern tuples according to Theorem 4 and Proposition 3.*

Figure 3 represent the simplified X-complete partitions lattice of the relation of Figure 1.

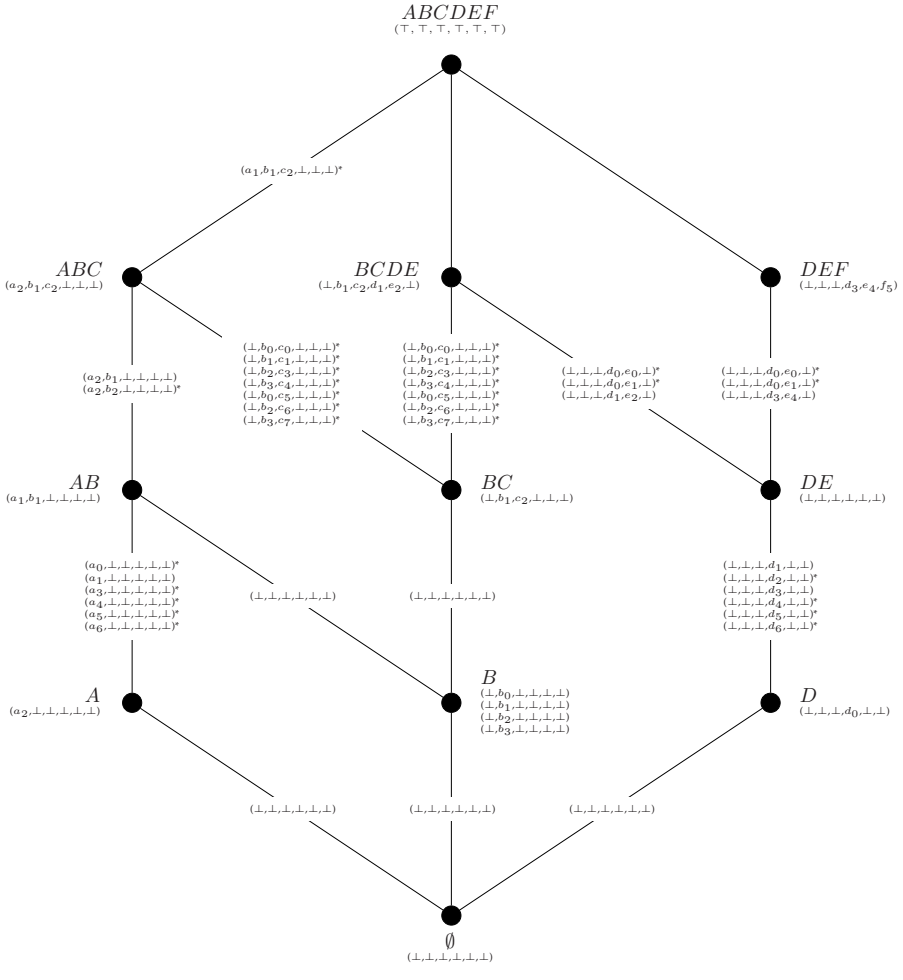


Fig. 3. The simplified X-complete partitions lattice of relation r. Pattern tuples marked with "*" select single tuples.

4 Link with FCA

4.1 Nominal Scaling and Global Concepts

A concept lattice is defined over a binary relation while the labeled lattice is defined over a many-valued relation. Some scaling has to be done in order to compare both lattices. Since our intersection operator on tuples is defined using the equality of values, the scaling we use is the classical *nominal scaling* of a many-valued relation: to each combination (attribute, value of attribute) corresponds a column in the resulting binary relation. Each line correspond to a tuple of the many-valued relation. An entry of the relation is equal to 1 if the corresponding tuple has the corresponding value for the given attribute in the many-valued relation. Otherwise, the entry equals to 0 (see [14] for more details on nominal scaling).

Given a many-valued relation r, we denote by $r_{(0,1)}$the corresponding binary relation obtained by a nominal scaling.

Note that we could use a *class equivalence* operator to define our intersection operator on tuples: in this case, the nominal scaling should be considered over equivalence classes. One can easily check the following property.

Property 3. A fragment relation $r' \subseteq r$ is a maximal X-complete relation such that $\theta(X, r') = X$ if and only if $(\gamma(X, r'), r')$ is a concept of $r_{(0,1)}$.

Previous property states that there is a one-to-one correspondence between X-complete relations which are X-closed and concepts of the scaled relation: $\gamma(X, r')$ is the intent while the list of tuples in the X-complete relation correspond to the extent. In the remaining of this paper, we thus use the notation $(\gamma(X, r'), r')$ to design formal concepts of $r_{(0,1)}$. According to the definition of $T_p(X)$, this leads immediately to the following property.

Property 4. Given a concept $(\gamma(X, r'), r')$ of $r_{(0,1)}$, there exists a closed set X of r such that $\gamma(X, r') \in T_p(X)$.

In other words, a concept of $r_{(0,1)}$is an *instance* of a closed set of r. Closed sets of the labeled lattice can thus be seen as *global concept* of the many-valued relation, formal concepts being just particular cases of such global concepts. Note that $T_p(X)$ might be empty (for instance, for the closed set DE in our example). In this case, no formal concept corresponds to this global concept. Consequence is that for those global concepts X with empty tableaux, any combination (present in the relation) of values of X will be the antecedent of an exact AR in the scaled relation.

Property 5. Let X be a closed set of r such that $T_p(X)$ is empty. Then, for any tuple $t \in r$, there exists $A \in Attr(R) \setminus X$ such that $t[X] \rightarrow t[A]$ is an exact association rule of r.

On the other extremity, a global concept X might have a tableau which selects the entire many-valued relation. In this case, any combination of values of X defines a formal concept in $r_{(0,1)}$. Such closed sets are called *strong global concepts*.

Property 6. Let X be a closed set of r such that $r_{T_p(X)} = r$. Then for any tuple $t \in r$, $t[X]$ is a closed set in $r_{(0,1)}$and t belongs to the extent of the formal concept defined by $t[X]$.

Note that given a strong concept X and for any attribute $A \notin X$, $T_p(X \rightarrow A)$ is empty. In other words, there does not exist any exact association rule which antecedent is a combination of values of X. This leads to the definition of the *strength* of a global concept.

Definition 8. *Given a global concept* $(X, T_p(X))$*, the* strength *of* X*, denoted by* $strength(X)$*, is the number of formal concept of* $r_{(0,1)}$ *which are instances of* X *divided by the total number of combination of values of* X *present in the relation. More formally:*

$$strength(X) = \frac{\mid T_p(X) \mid}{\mid \Gamma(X, r) \mid}$$

The strength of a global concept is an indicator which does not take into account the support of a formal concept. Another indicator could be also the *frequency* of a concept, which takes into account how many tuples verify a given concept.

Definition 9. *Given a global concept* $(X, T_p(X))$*, the* frequency *of* X*, denoted by* $freq(X)$ *is the probability, given a tuple* $t \in r$*, that* $t[X]$ *is a formal concept of* $r_{(0,1)}$*:*

$$freq(X) = \frac{\mid r_{T_p(X)} \mid}{\mid r \mid}$$

4.2 Association Rules and Approximate Association Rules

In [17], we have shown the hierarchy existing among ARs and CFDs, and also among approximate ARs and approximate FDs. The following properties show the link between ARs and edges of the labeled lattice and approximate ARs and global concepts.

Property 7. Given a global concept $(X, T_p(X))$. Then, for any tuple t in r such that $t(X) \in T_p(X)$ and for any attribute $A \notin X$, the implication $t[X] \rightarrow t[A]$ is an approximate AR.

In other words, a global concept is the union of approximate ARs. If the tableau of a global concept is empty, this global concept defines only exact ARs.

Theorem 5 (ARs computation). *The set of all exact ARs can be computed from the tableaux associated to edges of the labeled lattice.*

Proof. Immediate, from Theorems 1 and 2. □

In other words, a CFD is the union of exact ARs. If the tableau of an edge $X \rightarrow A$ is empty, then for any tuple t of r, $t[X] \rightarrow t[A]$ is an approximate AR.

We have seen in the previous section (theorem 4) that given $X \subset Y$ and for any $A \notin Y$ we have $r_{T_p(X \rightarrow A)} \subseteq r_{T_p(Y \rightarrow A)}$. The labeling of the lattice could thus

be simplified by labeling the edge $Y \rightarrow A$ with the pattern tuples of $T_p(Y \rightarrow A)$ which select tuples in $r_{T_p(X \not\rightarrow A)}$. We have seen that this labeling removes redundancy.

In a similar way, Proposition 3 shows that some pattern tuples in the tableau of a CFD $X \rightarrow Y$ play the role of *local keys* in the nominal scaling of the relation: the pattern tuple selects a single tuple in the relation. This tuple will thus appear in the tableaux of all CFDs which antecedent contains X: again this corresponds to some redundancy that could be removed from the labeling of the lattice. In the Figure 3, pattern tuples which select a single tuple are marked with a "*": such pattern tuples should thus appear only once in the lattice.

The simplified labeled lattice is thus the X-complete partition lattice with some redundancy removed in the labeling. This raises the following question.

Question 1 (Non redundant ARs). Does the set of ARs defined by the tableaux of the simplified labeled lattice define a non-redundant basis of ARs of the relation.

In other words, do we have removed all redundancy in the simplified labeled lattice ? This is still an open question for the moment.

Should the answer to the question be positive, this would give us a depth first search and pruning strategy to generate frequent non redundant AR. Indeed, given a closed set X and an attribute $A \notin X$, we know that $T_p(X \rightarrow A)$ defines ARs of the form $X \rightarrow A$. New non redundant ARs of the form $Y \rightarrow A$ would then be found in $r_{T_p(X \not\rightarrow A)}$. Thus, searching for these new non redundant ARs should be done only in this fragment relation: this reduces the size of the relation where to search for new ARs. Moreover, if this fragment relation has a size lower than the minimal desired support we would already know that no new *frequent* AR of the form $Y \rightarrow A$ could be found in it: we could stop the depth first search.

4.3 Dealing with Binary Relations

Traditional FCA considers only positive values of the binary relation. To obtain a labeled lattices which is isomorphic to the traditional concept lattice, it is thus sufficient to consider only positive values by replacing negative values with $\perp$. Note that in this case, the labels of all edges of the obtained lattice are empty and the tableaux of the closed sets contains only positive values. If both positive and negative values are considered in the labeled lattice, the CFDs and FDs obtained represent a basis of positive, negative and mixed association rules of the relation. In [18], mixed ARs are inferred from positive and negative ARs.

5 Discussion and Conclusion

In this paper we have presented the lattice of X-complete partitions of a many-valued relation. We have shown the link between this lattice and the concept lattice of the nominal scaling of the relation. Formal concepts are instances of closed sets in the labeled lattice which in turn can be seen as the union of several formal concepts. The labeled lattice can thus be seen as a *synthetic* representation

of all formal concepts holding in the relation. Closed sets of this lattice could thus be seen as global concepts of the relation. Moreover, edges of this labeled lattice represent a cover of CFDs holding in the many-valued relation. As a consequence, this lattice is also a synthetic representation of CFDs and ARs.

It should be noted that the aim of this paper is to present a structural framework which represents classical objects of FCA (at the value level) and to "compile" those objects at the attribute level. No algorithms were provided to construct this lattice of X-complete partitions or CFDs using this lattice. Interest of generating this lattice is not clear for the moment; main interest of this lattice is to understand the underlying properties of CFDs and global concepts.

Some authors have proposed extensions of FCA by taking into account some semantic in the values of attributes [8,13,15]. In all those cases, the obtained concept lattice define formal concepts at the value level. One should note that the lattice of X-complete partitions of a many-valued relation could be used to work at the attribute level for such extensions of FCA. A global concept (defined at the attribute level) is the merging of formal concepts (defined at the values level and taking into account some form of semantics).

We have presented several properties of this labeled lattice, such as the monotony existing on the fragment relation on which CFDs implying the same attribute hold. Given a CFD $X \rightarrow A$, this monotony property leads naturally to a search strategy for finding new informative CFDs of the form $Y \rightarrow A$ (with $X \subset Y$): these new CFDs are to be searched in the fragment relation on which the CFD $X \rightarrow A$ does not hold.

Next steps in our investigations is to answer the open question stated in this paper: does the set of ARs defined by the tableaux of the simplified lattice define a non redundant basis of ARs of the relation ? Should the answer be negative, this would indicate that some form of redundancy still remain in the simplified labeled lattice and thus, a more simplified labeling should be proposed. Should the answer to the question be positive, this would give us a direct search and pruning strategy to find a non redundant basis of ARs in the relation.

Other dependencies such as multivalued dependencies (MVD) where defined on many-valued relations. In [2], a closure operator for computing such dependencies as well as a lattice representation of those dependencies is proposed. An interesting question would be to investigate if such dependencies could be inferred directly in our lattice of X-complete partitions. Do some global concepts of our lattice represent antecedents of MVDs ?

Another interesting lead is to consider other scaling for the many-valued relation (see [14] for some other possible scalings). The lattice of X-complete partitions can be seen as the merging of formal concepts of the concept lattice in a single closed set. Different scalings could lead to other concept lattices (the chosen scaling introducing more semantics). Could the obtained formal concepts be merged in a global concept defined over attributes rather than values of attributes ? Our guess is that this merging operation is always possible (eventually by generating "empty" global concepts as it is the case with nominal scaling).

However, could such merging be explained in terms of X-complete partitions ? Will such scalings preserve the hierarchy between ARs and FDs ?

Acknowledgements

The authors wish to express their grateful acknowledgement to the anonymous referees for their constructive remarks and comments which helped improved clarity of the paper.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB, pp. 487–499 (1994)
2. Baixeries, J., Balcàzar, J.-L.: A lattice representation of relations, multivalued dependencies and armstrong relations. In: ICCS 2005, International Conference on Conceptual Structures (2005)
3. Baudinet, M., Chomicki, J., Wolper, P.: Constraint-generating dependencies. J. Comput. Syst. Sci. 59(1), 94–115 (1999)
4. Beeri, C., Vardi, M.Y.: Formal systems for tuple and equality generating dependencies. SIAM J. Comput. 13(1), 76–98 (1984)
5. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of armstrong relations for functional dependencies. Journal of the ACM 31, 30–46 (1984)
6. Bell, S., Brockhausen, P.: Discovery of data dependencies in relational databases. Technical Report LS-8 Report-14, University of Dortmund (1995)
7. Bohannon, P., Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for data cleaning. In: ICDE, pp. 746–755 (2007)
8. Chaudron, L., Maille, N.: Generalized formal concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 357–370. Springer, Heidelberg (2000)
9. Chomicki, J., Marcinkowski, J.: On the computational complexity of minimal-change integrity maintenance in relational databases. In: Bertossi, L., Hunter, A., Schaub, T. (eds.) Inconsistency Tolerance. LNCS, vol. 3300, pp. 119–150. Springer, Heidelberg (2005)
10. Codd, E.F.: Further normalizations of the database relational model. In: Rustin, R. (ed.) Data Base Systems, pp. 33–64. Prentice-Hall, Englewood Cliffs (1972)
11. De Bra, P., Paredaens, J.: An algorithm for horizontal decompositions. Inf. Process. Lett. 17(2), 91–95 (1983)
12. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for capturing data inconsistencies. ACM Trans. Database Syst. 33(2) (2008)
13. Ferré, S., Ridoux, O.: A logical generalization of formal concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 371–385. Springer, Heidelberg (2000)
14. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical foundations. Springer, Heidelberg (1999)
15. Gugish, R.: Many-valued context analysis using descriptions. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 157–168. Springer, Heidelberg (2001)

16. Huhtala, Y., Kärkkäinen, J., Porkka, P., Toivonen, H.: Tane: An efficient algorithm for discovering functional and approximate dependencies. The Computer Journal 42(2), 100–111 (1999)
17. Medina, R., Nourine, L.: A unified hierarchy for functional dependencies, conditional functional dependencies and association rules. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 98–113. Springer, Heidelberg (2009)
18. Missaoui, R., Nourine, L., Renaud, Y.: Generating positive and negative exact rules using formal concept analysis: Problems and solutions. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 169–181. Springer, Heidelberg (2008)
19. Petit, J.-M., Toumani, F., Boulicaut, J.-F., Kouloumdjian, J.: Towards the reverse engineering of denormalized relational databases. In: ICDE, pp. 218–227. IEEE Computer Society, Los Alamitos (1996)
20. Weddell, G.E.: Reasoning about functional dependencies generalized for semantic data models. ACM Transactions on Database Systems 17(1), 32–64 (1992)

Constrained Closed Datacubes

Sébastien Nedjar, Alain Casali, Rosine Cicchetti, and Lotfi Lakhal

Laboratoire d'Informatique Fondamentale de Marseille (LIF),
CNRS UMR 6166, Aix-Marseille Université
I.U.T. d'Aix en Provence, Avenue Gaston Berger,
13625 Aix en Provence Cedex1, France
firstname.lastname@lif.univ-mrs.fr

Abstract. This paper focuses on borders and lossless representations for Constrained Datacubes of database relations, which can represent many-valued contexts. The final goal is to optimize both storage space and computation time. First we study the succinct representation through the borders *Lower / Upper* and *Upper*$^\sharp$ */ Upper*. However, these borders are not information-lossless. Therefore, by using the concept of cube closure, from a FCA perspective, we define three new information lossless representations for Constrained Datacubes: the L-Constrained Closed Datacube, the $U^\sharp$-Constrained Closed Datacube and the $U^{\sharp\sharp}$-Constrained Closed Cube. The latter is obtained by using the constrained closed tuples along with the border *Upper*$^\sharp$ from which redundancy is discarded in order to obtain an optimal representation. Finally, we evaluate experimentally the size of the introduced representations. The results are strongly emphasizing the idea that the latest structure is, to the best of our knowledge, the smallest representation of Constrained Datacubes.

1 Introduction and Motivation

The Datacube [1] is a key concept for data warehouse management. Precomputing all the possible aggregates at various levels of granularity makes it possible to handle Datacubes and efficiently answer Olap queries. Several variations of the concept of Datacubes can be found in the recent research. The Constrained (Convex) Datacube concept was introduced in [2] to unify the Datacubes which are computed and pruned with monotone and/or anti-monotone constraints [3, 4]. For instance, Iceberg Datacubes are partial Datacubes inspired by frequent patterns. They capture only sufficiently significant trends by enforcing minimal threshold constraints over measures [5]. Window Datacubes can be seen as extending the previous ones by restricting measures to a given range [2]. In the latest context, users are provided with trends fitting in a particular "window". Emerging Cube [6–8] captures trends growing significant in a time frame. Dually, in the same time reference, the Emerging Cube emphasizes the vanishing trends. This kind of knowledge is central to multidimensional analysis of Olap.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 177–192, 2010.

Being a convex space the Constrained Datacube can be represented using one of the following couple of borders: (*i*) either the Lower and Upper borders [2], (*ii*) or the Upper$^\sharp$ and Upper borders [7]. The choice of borders addresses different issues:

(*i*) when determining if an unknown tuple belongs to the Constrained Datacube, the borders are used as a minimal compact representation of the Constrained Datacube; and
(*ii*) Nedjar [9] proposes a method based on the borders and on a probabilistic counting for calibrating the thresholds (linked to a constraint). As a consequence, we can nearly determine the size of a Constrained Datacube, without computing it.

However, not any couple of borders provide an information lossless representation because the associated measure values can no longer be derived. Often, the size of the Constrained Datacube is huge. A sound structure for reducing its size is necessary.

This paper focuses on information lossless representations of the Constrained Datacube which optimize both storage space and computation time. The main contributions are twofold:

(*i*) in retrieving the associated measure value: using the concept of cube closure [10], we propose three information-lossless representations: the L-Constrained Closed Datacube, the $U^\sharp$-Constrained Closed Datacube and the $U^{\sharp\sharp}$-Constrained Closed Datacube. These structures contain the borders $(L; U)$, $(U^\sharp; U)$ and $(U^{\sharp\sharp}; U)$ respectively. $U^{\sharp\sharp}$, called the *Reduced Upper*$^\sharp$ border, is achieved by discarding all the redundancies from $U^\sharp$. We show that the border $U^{\sharp\sharp}$ is the smallest information to be appended to the constrained closed tuples in order to achieve a lossless representation. We emphasize that these innovative structures can be easily adapted to the binary context. The reduced binary representation is also new.
(*ii*) in comparing the size of the borders L, $U^\sharp$ and $U^{\sharp\sharp}$: We perform experiments using classical data sets in data mining and data warehousing. Interesting results yield: when compared to L, $U^\sharp$ brings a systematic gain with a factor varying from 1.35 to 275 and the border $U^{\sharp\sharp}$ is obviously more compact.

The article is organized as follows. In Section 2, we resume the characterization of the Constrained Datacube and its representations with borders. Section 3 is devoted to the cube closure operator. The three information lossless representations based on the cube closure are detailed in Section 4 and the experiments are described in Section 5.

2 Constrained Datacube Framework

Let us consider a database relation r (which can represent a many-valued context [11]) with a set of attributes dimensions $\mathcal{D}$ (denoted by D_1, D_2, ..., D_n) and a set of measures (noted $\mathcal{M}$). The Constrained Datacube characterization fits

in the more general framework of the cube lattice of the relation r: $CL(r)$ [12]. The latter is a suitable search space which is to be considered when computing the Datacube of r. It organizes the tuples, possible solutions of the problem, according to a generalization / specialization order, denoted by $\preceq_g$ [13]. These tuples are structured according to the attributes dimensions of r which can be provided with the value ALL [1]. Moreover, we append to these tuples a virtual tuple which only encompasses empty values in order to close the structure. Any tuple of the cube lattice generalizes the tuple of empty values. For handling the tuples of $CL(r)$, the operator + is defined. This operator is a specification of the meet operator applied to the cube lattice framework [12]: provided with a couple of tuples, it yields the most specific tuple in $CL(r)$ which generalizes the two operands. Dually, the Product operator, noted $\bullet$, is a specification of the join operator in our framework. the Product of two tuples yields the most general tuple which specializes the two operands.

Example 1. Let us consider the relation DOCUMENT (*cf.* Table 1) giving the quantities of books sold by Type, City, Publisher and Language. In CL(DOCUMENT), let us consider the sales of Novels in Marseilles, whatever the publisher and language are, *i.e.* the tuple (Novel, Marseilles, ALL, ALL). This tuple is specialized by the two following tuples of the relation: (Novel, Marseilles, Collins, French) and (Novel, Marseilles, Hachette, English). Furthermore, (Novel, Marseilles, ALL, ALL) $\preceq_g$ (Novel, Marseilles, Collins, French) exemplifies the generalization order between tuples. Moreover we have (Novel, Marseilles, Hachette, English) + (Novel, Marseilles, Collins, French) = (Novel, Marseilles, ALL, ALL), and (Novel, Marseilles, ALL, French) $\bullet$ (ALL, Marseilles, Collins, ALL) = (Novel, Marseilles, Collins, French).

Table 1. Relation example DOCUMENT

Type	City	Publisher	Language	Quantity
Novel	Marseilles	Collins	French	400
Novel	Marseilles	Hachette	English	100
Textbook	Paris	Hachette	French	100
Textbook	Marseilles	Collins	English	300
Essay	Paris	Collins	French	200

Definition 1 (Measure function). *Let f be an aggregation function (*SUM, COUNT, MIN, MAX *...), r a database relation and t a tuple (or cell) of $CL(r)$. We denote by $f_{val}(t, r)$ the value of the aggregation function f associated to the tuple t in $CL(r)$.*

Example 2. If we consider the Novel sales in Marseilles, for any Publisher and Language, *i.e.* the tuple (Novel, Marseilles, ALL, ALL) of CL(DOCUMENT) we have: SUM_{val}((Novel, Marseilles, ALL, ALL), DOCUMENT) = 500.

In the remainder of this section, we study the cube lattice structure faced with conjunctions of monotone and anti-monotone constraints according to the generalization order. We show that this structure is a convex space which is called Constrained Datacube. We propose condensed representations (with borders) of the Constrained Datacube with a twofold objective: defining the solution space in a compact way and deciding whether a tuple t belongs or not to this space.

We take into account the monotone and anti-monotone constraints the most used in database mining [4]. They are applied on:

- measures of interest like pattern frequency, confidence, correlation. In these cases, only the dimensions of $\mathcal{D}$ are necessary;
- aggregates computed from measures of $\mathcal{M}$ and using statistic additive functions (SUM, COUNT, MIN, ...).

We recall the definitions of convex space, monotone and/or anti-monotone constraints according to the generalization order $\preceq_g$.

Definition 2 (Convex Space). *Let $(\mathcal{P}, \leq)$ be a partial ordered set, $\mathcal{C} \subseteq \mathcal{P}$ is a convex space if and only if $\forall x, y, z \in \mathcal{P}$ such that $x \leq y \leq z$ and $x, z \in \mathcal{C}$ then $y \in \mathcal{C}$. Thus $\mathcal{C}$ is bordered by two sets: (i) an "Upper set", noted U, defined by $U = \max_{\leq}(\mathcal{C})$, and (ii) a "Lower set", noted L and defined by $L = \min_{\leq}(\mathcal{C})$.*

Definition 3 (Monotone/anti-monotone constraints).

1. *A constraint Const is monotone according to the generalization order if and only if: $\forall t, u \in CL(r) : [t \preceq_g u$ and $Const(t)] \Rightarrow Const(u)$.*
2. *A constraint Const is anti-monotone according to the generalization order if and only if: $\forall t, u \in CL(r) : [t \preceq_g u$ and $Const(u)] \Rightarrow Const(t)$.*

Notations: We note *cmc* (*camc* respectively) a conjunction of monotone constraints (anti-monotone respectively) and *chc* an hybrid conjunction of constraints.

Example 3. In the multidimensional space of our relation example DOCUMENT (*cf.* Table 1), we would like to know all the tuples for which the measure value is greater than or equal to 200. The constraint "SUM(*Quantity*) $\geq$ 200" is anti-monotone. If the amount of sales by Type, City and Publisher is greater than 200, then the quantity satisfies this constraint at a more aggregated granularity level *e.g.* by Type and Publisher (all the cities merged) or by City (all the types and publishers together). In a similar way, if we aim to know all the tuples for which the quantity is lower than 400, the underlying constraint "SUM(*Quantity*) $\leq$ 400" is monotone.

Theorem 1. *The cube lattice with monotone and/or anti-monotone constraints (const) is a convex space which is called* ***Constrained Datacube****, $CD(r) = \{t \in CL(r)$ such that $const(t)\}$. Any tuple belonging to the Constrained Datacube is called a* ***constrained tuple****.*

In [2] we give different variants of Datacubes and their characterization as Constrained Datacube. For each one, we give the SQL query which computes it.

2.1 Borders [L;U]

The characterization of the Constrained Datacube as a convex space makes it possible to know whether a tuple satisfies or not the constraint conjunction by only knowing the classical lower and upper borders $[L;U]$ [2] of the Constrained Datacube. Actually if a conjunction of anti-monotone constraints holds for a tuple of $CL(r)$ then any tuple generalizing it also respects the constraints. Dually if a tuple fulfills a monotone constraint conjunction, then all the tuples specializing it also satisfy the constraints.

Definition 4 (Borders $[L;U]$). *The Constrained Datacube can be represented by the borders: U which encompasses the maximal constrained tuples and L which contains all the minimal constrained tuples according to $\preceq_g$.*

$$\begin{cases} L = \min_{\preceq_g}(\{t \in CL(r) \mid cmc(t) \wedge camc(t)\}) \\ U = \max_{\preceq_g}(\{t \in CL(r) \mid cmc(t) \wedge camc(t)\}) \end{cases}$$

Proposition 1. *The borders $[L;U]$ are a condensed representation for the Constrained Datacube: $\forall\ t \in CL(r)$, t is a constrained tuple if and only if $\exists (l,u) \in (L,U)$ such that $l \preceq_g t \preceq_g u$. In other words, t is a constrained tuple if and only if it belongs to the "range" [L; U].*

Table 2. Border U of the Constrained Datacube

(Essay, Paris, Collins, French)
(ALL, ALL, Hachette, ALL)
(Textbook, Marseilles, Collins, English)
(Novel, Marseilles, Collins, French)

Table 3. Border L of the Constrained Datacube

(ALL, ALL, Hachette, ALL)
(Essay, ALL, ALL, ALL)
(ALL, Paris, ALL, ALL)
(Textbook, ALL, ALL, ALL)
(ALL, ALL, ALL, English)
(ALL, Marseilles, ALL, French)
(Novel, ALL, Collins, ALL)
(Novel, ALL, ALL, French)

Example 4. With our relation example DOCUMENT, Tables 3 and 2 gives the borders $[L;U]$ for the Constrained Datacube provided with the constraints SUM(*Quantity*) $\in [200, 400]$. Provided with the borders, we know that the tuple (Essay, ALL, Collins, French) is a constrained tuple because it specializes the tuple (Essay, ALL, ALL, ALL) which belongs to the border L while generalizing the tuple (Essay, Paris, Collins, French) of the border U. Furthermore, the tuple (ALL, Marseilles, Hachette, ALL) is not a constrained tuple. Even if it specializes the tuple (ALL, ALL, Hachette, ALL) which belongs to the border L, it does not generalize any tuple of the border U.

2.2 Borders [U♯;U]

In this section, we present another condensed representation of the Constrained Datacube: the borders $[U^\sharp; U]$ [7]. This representation is based on the maximal tuples satisfying the anti-monotone constraint without verifying the monotone one.

Definition 5 (Borders $[U^\sharp; U]$). *The Constrained Datacube can be represented through two borders: U (cf. theorem 1) and $U^\sharp$ encompassing all the maximal tuples not satisfying the monotone constraint but satisfying the anti-monotone constraint. Thus, we have:*

$$\begin{cases} U^\sharp = \max_{\preceq_g}(\{t \in CL(r) \mid \neg cmc(t) \wedge camc(t)\}) \\ U = \max_{\preceq_g}(\{t \in CL(r) \mid cmc(t) \wedge camc(t)\}) \end{cases}$$

Example 5. With our relation example DOCUMENT, Table 4 gives the border $U^\sharp$ for the Constrained Datacube provided with the constraints SUM(*Quantity*) $\in [200, 400]$ (the border U is given in Table 2). For instance, the tuple (ALL, ALL, Collins, French) belongs to $U^\sharp$ because the number of French books published by Collins is greater than the given minimal threshold and all the tuples specializing (ALL, ALL, Collins, French), and satisfying the anti-monotone constraint, do not verify the monotone constraint. This is why the quoted tuple is maximal.

Table 4. Border $U^\sharp$ of the Constrained Datacube

(ALL, ALL, Collins, French)
(ALL, Marseilles, Collins, ALL)
(Novel, Marseilles, ALL, ALL)

With the following proposition, we are provided with a simple mechanism to know whether a tuple is a constrained tuple or not by using the borders $[U^\sharp; U]$.

Proposition 2. *The borders $[U^\sharp; U]$ are a condensed representation for the Constrained Datacube: $\forall\ t \in CL(r)$, t is a constrained tuple if and only if $\forall\ l \in U^\sharp$, $l \not\preceq_g t$ and $\exists u \in U$ such that $t \preceq_g u$. Thus t is a constrained tuple if and only if it belongs to the "range" $[U^\sharp; U]$.*

Example 6. With our relation example, the tuple (Essay, ALL, Collins, French) is a constrained tuple because it generalizes the tuple (Essay, Paris, Collins, French) which belongs to the border U and does not generalize any tuple of the border $U^\sharp$. Moreover (ALL, Marseilles, ALL, ALL) is not constrained because it generalizes the tuple (Novel, Marseilles, ALL, ALL) of the border $U^\sharp$.

3 Cube Closure

The cube connection [10] is a couple of functions $rc = (\lambda, \sigma)$, such that λ is defined from the cube lattice of r to the power set lattice of $Tid(r)$ and σ is the dual function of λ. We show that rc is a special case of Galois connection between two lattices [11]. Hence, we obtain a closure operator over $CL(r)$ under r.

Definition 6 (Cube Connection). *Let $Rowid : r \to \mathbb{N}^*$ be a mapping which associates each tuple with a single positive integer and $Tid(r) = \{Rowid(t)$ such that $t \in r\}$ (i.e. the set of the tuple identifiers of the relation r). Let λ and σ be two functions defined as follows:*

$\lambda : CL(r) \to \langle \mathcal{P}(Tid(r)), \subseteq \rangle$
$\quad t \mapsto \cup\{Rowid(t') \in Tid(r)$ *such that* $t \preceq_g t'$ *and* $t' \in r\}$
$\sigma : \langle \mathcal{P}(Tid(r)), \subseteq \rangle \to CL(r)$

$$P \mapsto \begin{cases} +\{t \in r \text{ such that } Rowid(t) \in P\} \\ (\emptyset, \ldots, \emptyset) \text{ otherwise.} \end{cases}$$

where $\mathcal{P}(Tid(r))$ stands for the power set of the tuple identifers of the relation r (Tid(r)).

Proposition 3. *The cube connection $rc = (\lambda, \sigma)$ is a Galois connection between the cube lattice of r and the power set lattice of $Tid(r)$.*

Definition 7 (Cube Closure). *Let $T \subseteq CL(r)$ be a set of tuples, the Cube Closure operator $\mathcal{C} : CL(r) \to CL(r)$ according to T can be defined as follows:*

$$\mathcal{C}(t, T) = \sigma \circ \lambda(t) = (\emptyset, \ldots, \emptyset) + \sum_{\substack{t' \in T, \\ t \preceq_g t'}} t'$$

where the operator $\sum$ has a very same semantics as the operator $+$.

Since $rc = (\lambda, \sigma)$ is a Galois connection, $\mathcal{C} = \sigma \circ \lambda$, is a closure operator [11].

Let us consider all the tuples t' in T. Let us aggregate them together by using the operator $\sum$. We obtain a new tuple which generalizes all the tuples t' and which is the most specific one. This new tuple is the closure of t.

Example 7. We achieve the closure of the tuple (Novel, ALL, ALL, ALL) in the relation DOCUMENT by aggregating all the tuples which specialize it by using the operator $+$. $\mathcal{C}$((Novel, ALL, ALL, ALL), DOCUMENT)= $(\emptyset, \ldots, \emptyset)$ + (Novel, Marseilles, Collins, French) + (Novel, Marseilles, Hachette, English) = (Novel, Marseilles, ALL, ALL).

Definition 8 (Measure function compatible with the cube closure). *A measure function, f_{val}, relative to an aggregate function f, from $CL(r) \to \mathbb{R}$ is compatible with the closure operator $\mathcal{C}$ over T if and only if $\forall t,\ u \in CL(r)$, it satisfies the three following properties:*

1. $t \preceq_g u \Rightarrow f_{val}(t,T) \geq f_{val}(u,T)$ *or* $f_{val}(t,T) \leq f_{val}(u,T)$,
2. $\mathcal{C}(t,T) = \mathcal{C}(u,T) \Rightarrow f_{val}(t,T) = f_{val}(u,T)$,
3. $t \preceq_g u$ *and* $f_{val}(t,T) = f_{val}(u,T) \Rightarrow \mathcal{C}(t,T) = \mathcal{C}(u,T)$.

This function is an adaptation of the weight function introduced in [14] for any closure system of the power set. For example the measure functions COUNT and SUM are compatible with the Cube Closure operator.

Thus in the same spirit as in [14], we can give another definition of the cube closure operator using the previous measure functions. The Cube Closure operator according to T can be defined as follows:

$$\mathcal{C}(t,T) = t \bullet \{t' \in Atom(CL(r)) : f_{val}(t,T) = f_{val}(t \bullet t', T)\}.$$

4 Structures of Constrained Closed Datacubes

The idea behind our representation is to remove redundancies existing within Constrained Datacubes. Actually certain tuples share a same semantics while others are more aggregated. In fact the ones and others are built up by aggregating the very same tuples of the original relation but at different granularity levels. Thus a single tuple, the most specific of them, can stand for the whole set. The Cube Closure operator is intended for computing this representative tuple.

Definition 9 (Constrained Closed Tuple). *Let $t \in CL(r)$ be a tuple, t is a constrained closed tuple if and only if :*

1. *t is a constrained tuple;*
2. *$\mathcal{C}(t,r) = t$.*

Of course the closure of any constrained tuple is a constrained closed tuple because, by its very definition, it is the most specific among all the tuples which generalize it. Thus it is necessarily equal to its own closure.

Example 8. The tuple (ALL, Marseilles, ALL, English) is a constrained closed tuple because:

1. (ALL, Marseilles, ALL, English) is a constrained tuple.
2. $\mathcal{C}$((ALL, Marseilles, ALL, English), DOCUMENT) = (ALL, Marseilles, ALL, English).

Unfortunately, the set of constrained closed tuples is not a lossless representation of the Constrained Datacube because for certain tuples it is not possible to decide whether they are constrained or not. They are all the tuples more general than the most general constrained closed tuples. For instance, let us consider the set of all constrained closed tuples (T) in Table 5. The tuples (ALL, Marseilles, Collins, French) and (ALL, ALL, Collins, French) share the same closure on T : (Novel, Marseilles, Collins, French) which is a constrained closed tuple. The former tuple is also constrained while the latter is not constrained.

In order to achieve a lossless representation, we combine on the one hand the set of constrained closed tuples from which the measure values can be retrieved and on the other hand the borders which delimit the space of solutions. However, the border U is already included in the closed tuple set, because the elements of U are the most detailed (specific) constrained tuples. Thus they are necessarily closed tuples.

4.1 *L*-Constrained Closed Datacubes

In this section, we introduce the **L**-Constrained Closed Cube which includes both *(i)* the set of constrained closed tuples and *(ii)* the lower border L. This approach is in the same spirit as the one proposed in [15] in the context of transaction databases and which encompasses the constrained closed patterns and the *Lower* border (L).

For reason of simplicity we use, from now on, t instead of $(t, f_{val}(t, r))$ to indicate a complete tuple with its dimension values and its measure.

Definition 10 (L-Constrained Closed Datacubes)
The L-Constrained Closed Datacube is defined as follows: $LCCD(r) = \{t \in CL(r)$ *such that* t *is a constrained closed tuple*$\} \cup L$.

Example 9. The L-Constrained Closed Datacube is represented through Table 5 giving the set of constrained closed tuples and Table 3 which proposes the lower border L.

Table 5. Set of constrained closed tuples

Constrained Closed Tuple	Sum(*Quantity*)
(Novel, Marseilles, Collins, French)	400
(Textbook, ALL, ALL, ALL)	400
(ALL, Marseilles, ALL, English)	400
(ALL, Paris, ALL, French)	300
(Textbook, Marseilles, Collins, English)	300
(ALL, ALL, Hachette, ALL)	200
(Essay, Paris, Collins, French)	200

In order to prove that the L-Constrained Closed Datacube is a lossless representation for the Constrained Datacube, we introduce a lemma. It shows that for any constrained tuple, we can compute its cube closure from either r or the L-Constrained Closed Datacube, and of course obtain the same result. Due to second equation of the measure function compatible with the cube closure, if two tuples have a similar cube closure, they have a similar value for the aggregative function.

Lemma 1. *For all constrained closed tuples t, $\mathcal{C}(t, LCCD(r)) = \mathcal{C}(t, r)$*

Proof. t is an constrained closed tuple then $\mathcal{C}(t,r) = (\emptyset, \ldots, \emptyset) + \sum t' = t$ such that $t' \in r$ and $t \preceq_g t'$. Thus $\mathcal{C}(t, LCCD(r)) = (\emptyset, \ldots, \emptyset) + \sum t'$ such that $t' = (\emptyset, \ldots, \emptyset) + \sum v$, $v \in r$, $t' \preceq_g v$ and $t \preceq_g t'$. Thus $\mathcal{C}(t, LCCD(r)) = (\emptyset, \ldots, \emptyset) + \sum \sum v$ such that $v \in r$ and $t \preceq_g v$. Due to the additivity property of $\sum$, we have $\mathcal{C}(t, LCCD(r)) = (\emptyset, \ldots, \emptyset) + \sum v = \mathcal{C}(t, r)$ with $v \in r$ and $t \preceq_g v$. □

The following proposition makes sure that the L-Constrained Closed Datacube is a lossless representation for the Constrained Datacube.

Proposition 4. *The L-Constrained Closed Datacube is a lossless representation for the Constrained Datacube: $\forall\ t \in CL(r)$, t is a constrained tuple if and only if $\mathcal{C}(t, LCCD(r)) \in LCCD(r)$.*

Proof. If t is constrained, we know that $t' = \mathcal{C}(t, r)$ has a measure equal to the one of t. Since t is constrained t' is also constrained. Thus t' is a constrained closed tuple according to proposition 1:

$$\mathcal{C}(t', r) = \mathcal{C}(t', LCCD(r)) = \mathcal{C}(t, LCCD(r))$$

Thus we have $\mathcal{C}(t, LCCD(r)) \in LCCD(r)$. □

If t is not constrained then $\nexists l \in L$ such that $l \preceq_g t$ and thus for all constrained closed tuple t' such that $t \not\preceq_g t'$. Since the closure of a tuple is the sum of all the tuples specializing it and t is not specialized by tuples of L then $\mathcal{C}(t, \{t' \in CL(r)$ such that t' is a constrained closed tuple$\})$ is not a constrained closed tuple. □

Example 10. For instance, let us derive the sum of quantity of books sold for the tuple (ALL, Marseilles, Collins, French). We know that this tuple is a constrained tuple because it is in the range $[L; U]$ (*cf.* Tables 3 and 2). By computing its cube closure over $LCCD$(Document), we obtain the tuple (Novel, Marseilles, Collins, French). Since the value of the Sum function of the previous tuple is 400, we make sure that the value of the Sum function of (ALL, Marseilles, Collins, French) is 400 and thus retrieve the expected result.

4.2 $U^\sharp$-Constrained Closed Datacubes

In this section, we introduce a new structure: the $U^\sharp$-Constrained Closed Datacubes. It includes both (i) the set of constrained closed tuples and (ii) the border $U^\sharp$. From definition 8, we can provide another characterization of the tuples of the border $U^\sharp$: they are closed tuples satisfying the anti-monotone constraint, but not the monotone constraint.

Definition 11 ($U^\sharp$-Constrained Closed Datacube). *The $U^\sharp$-Constrained Closed Datacube is defined as follows: $U^\sharp CCD(r) = \{t \in CL(r)$ such that t is a constrained closed tuple$\} \cup U^\sharp$.*

Example 11. The $U^\sharp$-Constrained Closed Datacube is represented through Table 5 giving the set of constrained closed tuples and Table 4 which proposes the upper border $U^\sharp$.

In order to prove that the $U^\sharp$-Constrained Closed Datacube is a lossless representation for the Constrained Datacube we introduce two propositions. The first one shows that for any constrained tuple, we can compute its cube closure from either r or the $U^\sharp$-Constrained Closed Datacube, and of course obtain the same result. The second one shows that two tuples having a same cube closure have a same measure.

Lemma 2. *For all constrained closed tuples t, $\mathcal{C}(t, U^\sharp CCD) = \mathcal{C}(t, r)$*

Proof. The proof is similar as the one of lemma 1 by replacing the border L by $U^\sharp$.

Proposition 5. *The $U^\sharp$-Constrained Closed Datacube is a lossless representation for the Constrained Datacube: $\forall\ t \in CL(r)$, t is a constrained tuple if and only if: $\mathcal{C}(t, U^\sharp CCD(r)) \in U^\sharp CCD(r) \backslash U^\sharp$.*

Proof. If t is constrained, we know that $t' = \mathcal{C}(t, r)$ has a measure equal to the one of t. Since t is constrained t' is also constrained. Thus t' is a constrained closed tuple, according to proposition 1 :

$$\mathcal{C}(t', r) = \mathcal{C}(t', U^\sharp CCD(r)) = \mathcal{C}(t, U^\sharp CCD(r))$$

Thus we have $\mathcal{C}(t, U^\sharp CCD(r)) \in U^\sharp CCD(r)$. □

If t is not constrained then $\exists u \in U^\sharp$ such that $t \preceq_g u$ and thus for all the constrained closed tuples t', $t \not\preceq_g t'$. Since the closure of a tuple is the sum of all the tuples specializing it and t is only specialized by tuples of $U^\sharp$ then $\mathcal{C}(t, U^\sharp CCD(r) \backslash U^\sharp) \notin U^\sharp CCD(r) \backslash U^\sharp$. □

Example 12. Let us derive the measure of (Novel, ALL, Collins, French). We know that this tuple is constrained because it is in the range $[U^\sharp; U]$ (*cf.* Tables 4 and 2). By computing its cube closure over $U^\sharp CCD(r)$(Document), we obtain the tuple (Novel, Marseilles, Collins, French). Since the value of the Sum function of the previous tuple is 400, we make sure that the value of the Sum function of (Novel, ALL, Collins, French) is 400 and thus retrieve the expected result.

4.3 $U^{\sharp\sharp}$-Constrained Closed Datacubes

In the previous section, we have shown that the border $U^\sharp$ must be appended to the constrained closed tuples in order to achieve an information-lossless representation of the Constrained Datacube. By making use of the cube closure, we simplify the border $U^\sharp$ by discarding all the redundancies that it can encompass. By this way, we obtain a new lossless representation: the $U^{\sharp\sharp}$-Constrained Closed Datacube.

Definition 12 (Redundant Closed Tuple). *For all tuple $t \in U^{\sharp}$, t is a redundant closed tuple if and only if:*

$$\mathcal{C}(t, U^{\sharp}CCD(r) \backslash \{t\}) = t$$

Let us notice that the above definition is proposed in the same spirit as the elimination of redundant attributes when computing minimal covers for functional dependencies.

Example 13. The tuple (ALL, Marseilles, Colins, ALL) is a redundant closed tuple because $\mathcal{C}$((ALL, Marseilles, Colins, ALL), $U^{\sharp}CCD$(Document)$\backslash\{$(ALL, Marseilles, Colins, ALL)$\}) = (\emptyset, \ldots, \emptyset)$ + (Novel, Marseilles, Collins, French) + (Textbook, Marseilles, Collins, English) = (ALL, Marseilles, Colins, ALL).

Definition 13 ($U^{\sharp\sharp}$ Border). *The Reduced Border $U^{\sharp}$ is defined as follows :*

$$U^{\sharp\sharp} = \{t \in U^{\sharp} \text{ such that } t \text{ is not a redundant closed tuple}\}$$

Example 14. With our example, the *Reduced* Border $U^{\sharp\sharp}$ encompasses a single tuple: (Novel, Marseilles, ALL, ALL). The other tuples of $U^{\sharp}$ are discarded (*cf.* definition 12).

Let us recall that, like all the tuples of $U^{\sharp}$, all the tuples of $U^{\sharp\sharp}$ satisfy the anti-monotone constraint. If we consider the lattice $\mathcal{L}$ which encompasses all the constrained closed tuples satisfying the anti-monotone constraint, the tuples of $U^{\sharp\sharp}$ are the meet-irreducible elements of $\mathcal{L}$ which do not verify the monotone constraint.

Definition 14 ($U^{\sharp\sharp}$-Constrained Closed Datacube). *The $U^{\sharp\sharp}$-Constrained Closed Datacube, noted $U^{\sharp\sharp}CCD$, is defined as follows: $U^{\sharp\sharp}CCD(r) = \{t \in CL(r)$ such that t is a constrained closed tuple$\} \cup U^{\sharp\sharp}$.*

Example 15. With our relation example, the $U^{\sharp\sharp}$-Constrained Closed Datacube is composed of the constrained closed tuples given in Table 5, and the border $U^{\sharp\sharp}$.

The following proposition shows that removing redundant closed tuples from the border $U^{\sharp}$ does not alter the closure computation for the constrained closed tuples. Thus the $U^{\sharp\sharp}$-Constrained Closed Datacube is a lossless representation for the $U^{\sharp\sharp}$-Constrained Closed Datacube and by transitivity it is a lossless representation for the Constrained Datacube (*cf.* proposition 6).

Lemma 3. $\forall\ t \in U^{\sharp\sharp}CCD(r)$, $\mathcal{C}(t, U^{\sharp\sharp}CCD(r)) = \mathcal{C}(t,\ U^{\sharp}CCD(r))$.

Proof. The proof is similar as the one of lemma 1 by replacing the border L by $U^{\sharp\sharp}$.

Proposition 6. *The $U^{\sharp\sharp}$-Constrained Closed Datacube is a lossless representation for the Constrained Datacube: $\forall\ t \in CL(r)$, t is a constrained tuple if and only if $\mathcal{C}(t, U^{\sharp\sharp}CCD(r))$ is a constrained closed tuple.*

Proof. The tuples discarded from $U^{\sharp}$ does not alter the closure computation for the constrained closed tuples because they are not Meet irreducible. □

Example 16. Let us derive the value of the SUM function of the tuple (Novel, ALL, Collins, French). We know that this tuple is possibly constrained because it generalizes the tuple (Novel, Marseilles, Collins, French) of U. By computing its cube closure over $U^{\sharp\sharp}CCD$(DOCUMENT), we obtain the tuple (Novel, Marseilles, Collins, French). Since this closed tuple is a constrained closed tuple, and its value for the measure is 400, we can assert that the value for the SUM of (Novel, ALL, Collins, French) is 400.

Concerning the tuple (ALL, Marseilles, ALL, ALL), we can say that this tuple is possibly a constrained tuple because it generalizes the tuple (Textbook, Marseilles, Collins, English) belonging to the border U. However, its cube closure over $U^{\sharp\sharp}CCD$(DOCUMENT), *i.e.* the tuple (ALL, Marseilles, ALL, ALL), does not belong to the set of contrained closed tuples (*cf.* Table 5), this is why we can say that (ALL, Marseilles, ALL, ALL) is not a constrained tuple.

5 Experimental Evaluations

In order to validate the lossless representations based on the cube closure (*cf.* section 4), we perform experiments to evaluate only the size of the borders *Lower* (L), *Upper*$^{\sharp}$ ($U^{\sharp}$) and *Reduced Upper*$^{\sharp}$ ($U^{\sharp\sharp}$). Remark that the constrained closed tuples remains the same over the three proposed representations. For this purpose, we use classical data sets[1,2]. We choose real and synthetic data sets. Their characteristics are reported in Table 6. They are:

- the data set of census PUMSB extracted from "PUMSB sample file",
- the real data set SEP85L containing weather conditions at various weather stations or lands from December 1981 to November 1991. This weather data set has been frequently used in calibrating various cube algorithms [13],
- the synthetic data sets T10I4D100K and T40I10D100K, built up from sale data.

Table 6. Data Sets

Name	#Tuples	#Attributes	#Values
PUMSB	49 046	74	2 113
SEP85L	1 015 367	9	7 175
T10I4D100K	100 000	10	1 000
T40I10D100K	100 000	40	1 000

[1] http://fimi.cs.helsinki.fi/
[2] http://cdiac.ornl.gov/ftp/ndp026b/SEP85L.DAT.Z

In our experiments, for computing the border $U^\sharp$, we choose the algorithm MAFIA [16] because of its efficiency and availability. In order to evaluate the size of the border L, we use the algorithm MCTR [17].

We choose to consider two distinct thresholds minimal ($MinThd$) and maximal (for the anti-monotone and monotone constraints). For any data set, the minimal threshold remains similar during all the experiments. This threshold has a value slightly lower than the lowest value of the maximal threshold. Moreover the latter varies in a range appropriated to each data set. By this way, we obtain significant results (neither too low nor too high). We propose a presentation both graphic and quantitative for the evaluation of the number of elements in the borders.

$MinThd$	$\lvert L\rvert$	$\lvert U^\sharp\rvert$	$\lvert U^{\sharp\sharp}\rvert$
90	2 693	259	0
80	11 876	3 145	0
70	39 905	11 737	0
60	131 179	37 388	0
55	409 167	92 221	0
50	1 052 671	193 939	8099

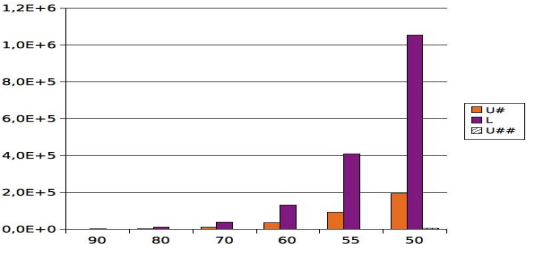


Fig. 1. Experimental Results for PUMSB

$MinThd$	$\lvert L\rvert$	$\lvert U^\sharp\rvert$	$\lvert U^{\sharp\sharp}\rvert$
0.1	20 020	3 506	0
0.08	25 256	4 298	0
0.04	33 235	8 958	0
0.02	70 474	20 078	0
0.01	182 218	43 282	9 084

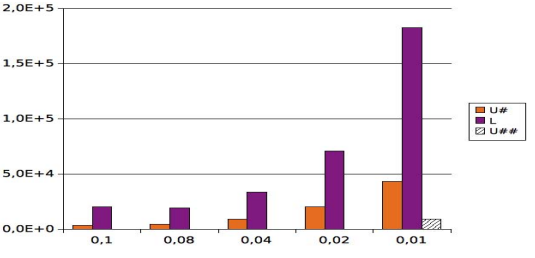


Fig. 2. Experimental Results for SEP85L

In any case, the border $U^\sharp$ is significantly more reduced than L. For instance, for real data sets, in the most critical cases when the threshold is very low and borders are large, the size of $U^{\sharp\sharp}$ is very small and of course more reduced than $U^\sharp$. By increasing the threshold, the border size obviously decreases. We establish that the gain factor provided by $U^\sharp$ when compared to L is even more important. In such cases, $U^{\sharp\sharp}$ has a null size which means that the $U^{\sharp\sharp}$-Constrained Closed Datacube only encompasses the constrained closed tuples. The experiments run on synthetic data confirm and enlarge these results with a gain factor varying from 76 to 275 (T40I10D100K) or 25 to 55 (T40I10D100K) for $U^\sharp$. As previously

$MinThd$	$\|L\|$	$\|U^{\sharp}\|$	$\|U^{\sharp\sharp}\|$
1.00	70 611	370	0
0.50	161 617	585	0
0.25	255 762	1 592	3
0.15	295 668	2 399	43
0.10	344 651	4 054	220
0.01	9 755 344	127 264	86 137

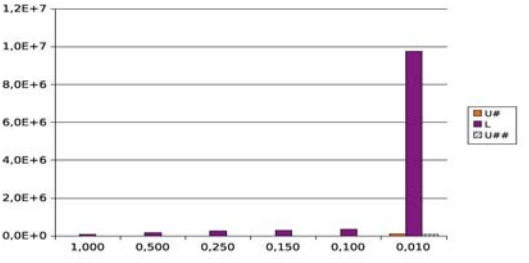


Fig. 3. Experimental Results for T10I4D100K

$MinThd$	$\|L\|$	$\|U^{\sharp}\|$	$\|U^{\sharp\sharp}\|$
1.5	270 289	4 947	0
1.0	519 679	21 217	0
0.8	792 785	27 066	9 473
0.5	2 165 949	65 815	41 193

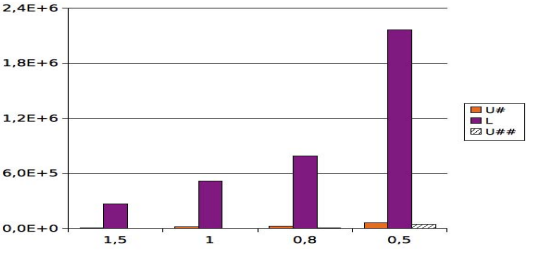


Fig. 4. Experimental Results for T40I10D100K

indicated, $U^{\sharp\sharp}$ has a size systematically null when the threshold is high. When compared to the size of $U^{\sharp}$, the gain factor is around 1.5 in the worse cases (when the two thresholds are very close).

6 Conclusion

The borders *Lower / Upper* and *Upper*$^{\sharp}$ */ Upper* are possible representations of a convex space, which is the case of any Datacube computed with monotone and/or anti-monotone constraints. However, this convex structure does not make possible to retrieve the value of the aggregative function of a tuple which belongs to the solution space. This is why, we have proposed three information lossless structures based on the cube closure operator [10]: the L-Constrained Closed Datacube, $U^{\sharp}$-Constrained Closed Datacube and the $U^{\sharp\sharp}$-Constrained Closed Datacube. The latter is built up from the $U^{\sharp}$-Constrained Closed Datacube, from which we have discarded the redundancies in the $U^{\sharp}$ border. Experimental evaluations comparing the size of the three mentioned representation have shown that the size of the $U^{\sharp\sharp}$-Constrained Closed Datacube is really smaller than the other ones.

An interesting prospect of the presented approaches is to find a method which, given the size of the Constrained Datacube [9], chooses the best algorithm for computing the Constrained Closed Datacube.

References

1. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. Data Min. Knowl. Discov. 1(1), 29–53 (1997)
2. Casali, A., Nedjar, S., Cicchetti, R., Lakhal, L.: Convex cube: Towards a unified structure for multidimensional databases. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 572–581. Springer, Heidelberg (2007)
3. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2006)
4. Pei, J., Han, J., Lakshmanan, L.V.S.: Pushing convertible constraints in frequent itemset mining. Data Min. Knowl. Discov. 8(3), 227–252 (2004)
5. Beyer, K.S., Ramakrishnan, R.: Bottom-up computation of sparse and iceberg cubes. In: Delis, A., Faloutsos, C., Ghandeharizadeh, S. (eds.) SIGMOD Conference, pp. 359–370. ACM Press, New York (1999)
6. Nedjar, S., Casali, A., Cicchetti, R., Lakhal, L.: Emerging cubes for trends analysis in olap databases. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 135–144. Springer, Heidelberg (2007)
7. Nedjar, S., Casali, A., Cicchetti, R., Lakhal, L.: Emerging cubes: Borders, size estimations and lossless reductions. Information Systems 34(6), 536–550 (2009)
8. Nedjar, S., Casali, A., Cicchetti, R., Lakhal, L.: Reduced representations of emerging cubes for olap database mining. IJBIDM 5(1), 268–300 (2010)
9. Nedjar, S.: Exact and approximate sizes of convex datacubes. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 204–215. Springer, Heidelberg (2009)
10. Casali, A., Nedjar, S., Cicchetti, R., Lakhal, L.: Closed cube lattices. Annals of Information Systems 3(1), 145–164 (2009); New Trends in Data Warehousing and Data Analysis
11. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
12. Casali, A., Cicchetti, R., Lakhal, L.: Cube lattices: A framework for multidimensional data mining. In: Barbará, D., Kamath, C. (eds.) SDM. SIAM, Philadelphia (2003)
13. Lakshmanan, L.V.S., Pei, J., Han, J.: Quotient cube: How to summarize the semantics of a data cube. In: Lochovsky, F.H., Shan, W. (eds.) VLDB, pp. 778–789. Morgan Kaufmann, San Francisco (2002)
14. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with titanic. Data Knowl. Eng. 42(2), 189–222 (2002)
15. Bonchi, F., Lucchese, C.: On closed constrained frequent pattern mining. In: Morik, K., Rastogi, R. (eds.) ICDM, pp. 35–42. IEEE Computer Society, Los Alamitos (2004)
16. Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., Yiu, T.: Mafia: A maximal frequent itemset algorithm. IEEE Trans. Knowl. Data Eng. 17(11), 1490–1504 (2005)
17. Casali, A., Cicchetti, R., Lakhal, L.: Extracting semantics from data cubes using cube transversals and closures. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) KDD, pp. 69–78. ACM, New York (2003)

Conceptual Navigation in RDF Graphs with SPARQL-Like Queries

Sébastien Ferré

IRISA, Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
ferre@irisa.fr

Abstract. Concept lattices have been successfully used for information retrieval and browsing. They offer the advantage of combining querying and navigation in a consistent way. *Conceptual navigation* is more flexible than hierarchical navigation, and easier to use than plain querying. It has already been applied to formal, logical, and relational contexts, but its application to the semantic web is a challenge because of inference mechanisms and expressive query languages such as SPARQL. The contribution of this paper is to extend conceptual navigation to the browsing of RDF graphs, where concepts are accessed through SPARQL-like queries. This extended conceptual navigation is proved *consistent* w.r.t. the context (i.e., never leads to an empty result set), and *complete* w.r.t. the conjunctive fragment of the query language (i.e., every query can be reached by navigation only). Our query language has an *expressivity* similar to SPARQL, and has a more *natural* syntax close to description logics.

1 Introduction

With the growing amount of available resources in the Semantic Web (SW), it is a key issue to provide an easy and effective access to them, not only to specialists, but also to casual users. The challenge is not only to allow users to retrieve particular resources (e.g., flights), but to support them in the exploration of a domain knowledge (e.g., which are the destinations? Which are the most frequent? With which companies and at which price?). We call the first mode *retrieval search*, and, following Marchionini [Mar06], the second mode *exploratory search*. The latter is generally supported by *faceted search* [ST09].

Conceptual navigation, based on Formal Concept Analysis (FCA) [GW99], also supports exploratory search by guiding users from concept to concept [CR96, DE08, Fer09]. The concept lattice plays the role of an exploration space, where each concept can be reached either by entering a query, or by following navigation links between concepts. At each step of the navigation, the set of navigation links is organized as a summary of the extent of the current concept, and provides insight and feedback about the context, and thus supports exploratory search. This solves the dilemma between using an expressive query language that is difficult to use (e.g., Boolean queries), and an intuitive but rigid navigation structure (e.g., file hierarchies).

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 193–208, 2010.

Languages of the SW, on the one hand, are more expressive than FCA, w.r.t. both the representation language (e.g., RDFS vs formal context) and the query language (e.g., SPARQL vs sets of attributes). Extensions of FCA such as Logical Concept Analysis (LCA) with relations [FRS05] or Relational Concept Analysis (RCA) [HHNV07] get closer to SW languages but each extension still misses large fragments of expressivity: e.g., LCA misses cycles in queries, RCA misses disjunction. On the other hand, querying languages for the SW (e.g., SPARQL [PAG06], OWL-QL [FHH04]), while expressive, are difficult to use, even for specialists, and do not provide enough feedback to satisfy exploratory search. Indeed, even if users have a perfect knowledge of the syntax and semantics of the query language, they may be ignorant about the application vocabulary, i.e., the *ontology*. If they also master the ontology or if they use a query assistant (e.g., Protégé[1]), the query will be syntactically correct and semantically consistent w.r.t. the ontology but can still produce no result (e.g., it makes sense to ask for a flight from Rennes to Agadir, but it happens there is none). Faceted search systems such as Slashfacet [HvOH06] or BrowseRDF [ODD06] rely on actual data instead of an ontology to assist users in their search. They do support exploratory search but with limited expressivity compared to SW query languages. For instance, they allow neither for cycles in queries, nor for general disjunction and negation.

The contribution of this paper is to adapt and extend conceptual navigation to the Semantic Web. We propose a navigation process that (1) is based on a query language whose *expressivity* is similar to SPARQL, and (2) has a *natural* and concise notation (similar to N3[2]), and that is (3) *consistent* (no dead-end) and (4) *complete* (every query can be reached by navigation). The last two points give a formal basis to conceptual navigation, and make it a real alternative, rather than a complement, to querying. Our approach builds upon, and is compatible with, existing techniques for designing and storing ontologies, reasoning, as well as querying languages and their implementations.

We first give basics of the Semantic Web (Section 2), and Logical Information Systems (Section 3) from which our extension starts. We then detail our proposal for conceptual navigation in the Semantic Web, separating the static part (user interface as a local view on the concept lattice, Section 4), and the dynamic part (user interactions as navigation links between concepts, Section 5). A few perspectives are discussed before concluding (Section 6).

2 Basics of the Semantic Web

The Semantic Web is founded on several representation languages, such as RDF, RDFS, and OWL, which provide increasing inference capabilities [HKR09]. The two basic units of these languages are *resources* and *triples*. A resource can be either a URI (Uniform Resource Identifier), a literal (e.g., a string, a number, a date), or a *blank node*, i.e., an anonymous resource. A URI is the absolute name of a *resource*, i.e., an entity, and plays the same role as URL

[1] See `http://protege.stanford.edu/`
[2] See `http://www.w3.org/DesignIssues/Notation3.html`

w.r.t. web pages. Like URLs, a URI can be a long and cumbersome string (e.g., `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`), so that it is often denoted by a qualified name (e.g., `rdf:type`). A triple (s, p, o) is made of 3 resources, and can be read as a simple sentence, where s is the subject, p is the verb (called the predicate), and o is the object. For instance, the triple `(ex:Bob,rdf:type,ex:man)` says that "Bob has type man", or simply "Bob is a man". Here, the resource `ex:man` is used as a class, and `rdf:type` is used as a property, i.e., a binary relation. The triple `(ex:Bob,ex:friend,ex:Alice)` says that "Bob has friend Alice", where `ex:friend` is another property. The triple `(ex:man,rdfs:subClassOf,ex:person)` says that "man is subsumed by person", or simply "every man is a person". The set of all triples of a knowledge base form a RDF graph. A RDF graph that uses the OWL vocabulary to define classes and properties is generally called an *ontology*.

RDF(S) introduces a vocabulary of resources to represent the membership to a class (`rdf:type`), subsumption between classes (`rdfs:subClassOf`) and between properties (`rdfs:subPropertyOf`), the domain (`rdfs:domain`) and range (`rdfs:range`) of properties, the meta-classes of classes (`rdfs:Class`) and of properties (`rdf:Property`), etc. OWL introduces additional vocabulary to represent complex classes and properties: e.g., restrictions on properties, intersection of classes, inverse property. The variant OWL-DL is the counterpart of Description Logics (DL) [BCM+03], where resources are individuals, classes are concepts, and properties are roles. Each language comes with a semantics, and the richer the vocabulary is, the more expressive and the more complex the inference is. In this paper, we do not make any strong assumption on the vocabulary.

Query languages provide on SW knowledge bases the same service as SQL on relational databases. They generally assume that implicit triples have been inferred and added to the base. The most well-known query language, SPARQL [PAG06], reuse the `SELECT FROM WHERE` shape of SQL queries, using graph patterns in the WHERE clause. For instance, twin siblings can be retrieved by the following query:

```
SELECT ?x ?y FROM <ex.rdf> WHERE { { ?x ex:mother ?z. ?y ex.mother
 ?z. ?x ex:birthdate ?d. ?y ex:birthdate ?d } FILTER (?x != ?y) }
```

Two persons `?x` and `?y` are twins if they share a same mother and a same birthdate, and are different. The FILTER condition is necessary because nothing prevents two variables to be bound to a same resource.

There exists a mapping from RCA to DL [HHNV07], or equivalently from RCA to OWL-DL. In short, it maps objects to resources, attributes to classes, the incidence relation to the property `rdf:type`, each context of a Relational Context Family (RCF) to a class, and each relation of the RCF to a property. Formal concepts are mapped to defined OWL classes, and subconcept links are mapped to the property `rdfs:subClassOf`. This suggests that the SW standard formats based on XML can be used to represent and share FCA data, both formal contexts and formal concept lattices. Therefore, every algorithm or system that works on SW data does work on FCA data. Conversely, a challenge for FCA is

to stretch its algorithms and systems so that they work on SW data [RKH07]. Previous work have mostly focused on the use of FCA to support the design of ontologies: e.g., implication basis for description logics [BD08], acquisition of OWL axioms based on attributed exploration [VR08]. The contribution of this paper is the extension of FCA-based conceptual navigation to the semantic web. We take Logical Information Systems (LIS) [Fer09] as a starting point because they share, at a lower level, expressive query languages and inference.

3 Basics of Logical Information Systems

We here recall the basics of Logical Information Systems (LIS) because we start from the structure of its user interface and interactions to conceptual navigation on RDF graphs. LIS instantiate both the conceptual navigation paradigm [Fer09], and the faceted search paradigm [ST09][3]. LIS user interface gives a *local view* of the concept lattice, centered on a concept called the *focus*. The local view is made of three parts: (1) the query, (2) the extent, and (3) the index. The query is a logical formula. The extent is the set of objects that are matched by the query, along the principles of logical concept analysis [FR04]. The extent identifies the focus concept. Finally, the index is a finite subset of the logic that is restricted to formulas that match at least one object in the extent. The index plays the role of a summary or inventory of the extent, showing which kinds of objects there are, and how many of each kind there are.

The query can be modified in three ways. To *query by formula* is to directly edit the query, which requires expertise or luck from the user. To *navigate* is to select formulas in the index in order to make the query more specific (moving downward in the lattice) or more general (moving upward in the lattice). To *query by examples* is to select a set of objects in the extent, which leads to the concept whose intent (the new query) is the conjunction of all properties shared by the selected objects. In the three cases, the modification of the query entails the update of the extent, hence updating the focus concept and the index. By definition of the index, no navigation link (a selection in the index) can lead to an empty result. Conversely, because the navigation structure is a lattice rather than a hierarchy, all valid conjunctions of formulas can be reached by navigation, and in any order. Contrary to querying by formula, navigation only requires from the user to recognize the meaning of formulas in the index, in the context of the application.

4 User Interface: Local View

The extension of the LIS framework to the semantic web applies to the query language and navigation modes, highly improving expressivity and flexibility when browsing a dataset. In this section, we redefine the LIS notion of *local view*. This comprises the definition of queries and their extents, and the summarization index over extents. Navigation links are defined in Section 5 on top of the local view.

[3] See Chapters 3, 4, 5, 8, and 9 of this book.

We consider the dataset to be a RDF graph [PAG06], which may include both explicit and implicit facts (i.e., triples) through reasoning. In this paper, we do not make the distinction between the two. In practice, the implicit facts may be materialized in a preprocessing stage, or infered on demand. We assume pairwise disjoint infinite sets of URIs (U), blank nodes (B), and literals (L). The set of *resources* is defined as $R = U \cup B \cup L$.

Definition 1 (RDF graph). *An* RDF graph *is defined as a set of* triples $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$*, where* s *is the* subject*,* p *is the* predicate*, and* o *is the* object*.*

RDF vocabulary for genealogy. For illustration purposes, we consider RDF graphs about genealogical data. The URIs of this domain are associated to a namespace `gen:`. This prefix is omitted if there is no ambiguity. Resources can be *persons*, *events*, *places* or literals such as names or dates. Persons belong either to the class of *men* or to the class of *women*, may have a *firstname*, a *lastname*, a *sex*, a *father*, a *mother*, a *spouse*, a *birth*, and a *death*. A birth or a death is an event that may have a *date* and a *place*. Places can be described as *parts* of larger places. OWL axioms may be used to enforce some invariants, e.g., the property *spouse* is symmetrical, the property *father* is functional, and the property *part* is transitive.

Figure 1 shows the user interface of our prototype, applied to the genealogy of George Washington[4]. It reflects the structure of a local view with the query at the top, the extent at the left, and the index at the center and right. The query selects "male persons whose lastname is Washington". There are 17 answers in the extent: e.g., George Washington. The central index shows that 7 of them have a known birth's place, and that 11 of them are known to be married. The right index shows their distribution in a taxonomy of locations, according to their birth's place. The hidden tabs give their distribution according to their birth's year or firstname.

4.1 Queries and Extensions

A SPARQL query can be used to define the focus of a local view. For instance, a local view that puts the focus on the set of "women whose some parent was born in Virginia in 1642" can be defined by the following SPARQL query.

```
SELECT ?x
WHERE { ?x rdf:type gen:woman. ?x gen:parent ?p. ?p gen:birth ?b.
        ?b gen:date 1642. ?b gen:place ?l. gen:VA gen:part ?l. }
```

In LIS, a query must necessarily define a set of resources, i.e., a mono-dimensional relation. This implies that we only need SPARQL queries with a single variable in the SELECT clause. This also means that our queries are analogous to OWL complex classes. In fact, the above query can be expressed in the description logic $\mathcal{SHOIN}$ that backs OWL-DL:

$$Woman \sqcap \exists parent.\exists birth.(\exists date.\{1642\} \sqcap \exists place.\exists part^{-}.\{VA\}).$$

[4] Downloadable at `http://www.irisa.fr/LIS/ferre/camelis/camelis2.html`

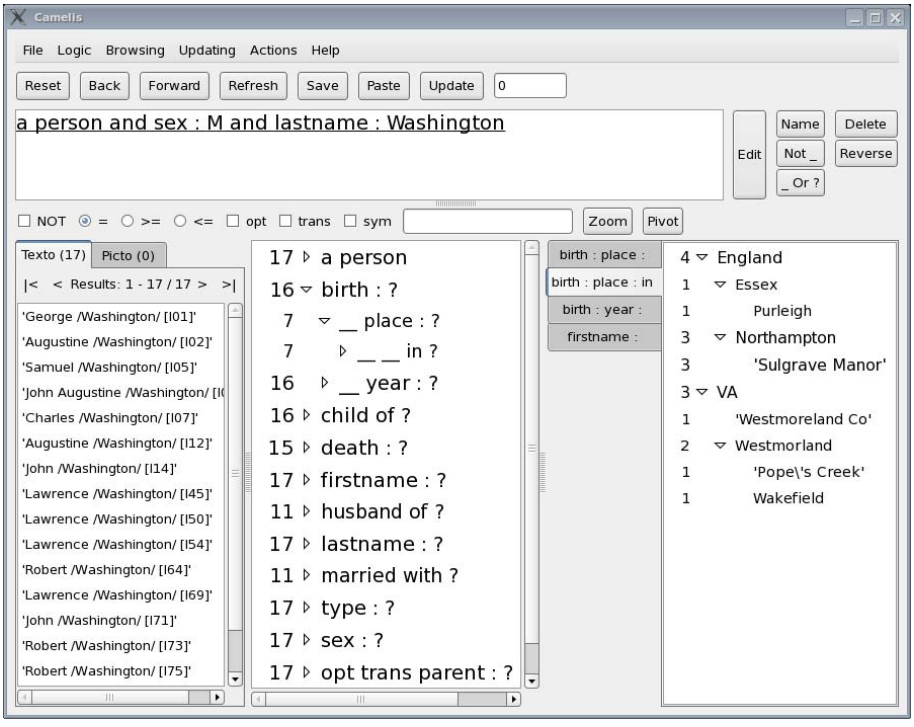


Fig. 1. A local view: query (top), extension (left), and index (center and right). The query selects male persons whose lastname is Washington.

The advantages of the DL syntax is that it is more concise, and that it avoids the use of variables. LIS do not require from end-users the ability to write queries, but they do require from them to understand queries. Ideally, the queries should be understandable with little, if any, learning. We think that the DL syntax is closer to this objective than the SPARQL syntax, provided that mathematical symbols are replaced by words, of course. At the same time, SPARQL has more expressive patterns (e.g., cycles).

We propose a new language for querying RDF graphs, where query results are sets of resources. Therefore, the expressions of this language are called *complex classes*, and make use of *complex properties*, derived from basic properties.

Definition 2 (complex property). *A* complex property *is any of:*

p `:`	*the property p itself,*
p `of`	*the inverse of the property p,*
p `with`	*the symmetric closure of the property p,*
`trans` P	*the transitive closure of the complex property P ("transitively P"),*
`opt` P	*the reflexive closure of the complex property P ("optionally P").*

Applying the three closures, `opt trans` p `with`, defines an equivalence relation, while `opt trans` P defines a partial ordering if P is antisymmetric, and a

pre-order otherwise. In the following, we use `in` as an abbreviation for the complex property `opt trans part of`.

Definition 3 (complex class). *Let V be an infinite set of variables, disjoint with the set of resources R. For every resource $r \in R$, variable $v \in V$, URI $u \in U$, complex property P, and complex classes C, C_1, C_2, the following expressions are also complex classes (in decreasing priority for operators):*

$$r \mid \texttt{?}v \mid \texttt{?} \mid \texttt{a}\ u \mid P\ C \mid \mathit{not}\ C_1 \mid C_1\ \texttt{and}\ C_2 \mid C_1\ \texttt{or}\ C_2.$$

Compared to DL languages, the complex class r corresponds to the *nominal* $\{r\}$, the anonymous variable `?` corresponds to the *top concept* $\top$, the expression $P\ C$ corresponds to a *qualified existential restriction* $\exists P.C$ (simply a *restriction* from now on), the expression `a` u corresponds to a *concept name*, the `and` corresponds to *concept intersection* $\sqcap$, the `or` corresponds to *concept union* $\sqcup$, and `not` corresponds to *concept complement* $\neg$. The addition of variables `?`v allows for the expression of cyclical graph patterns, like in SPARQL. The notation p `:` is reminiscent of the notation of valued attributes. For example, in the expression `name : "John"`, `name` is the attribute, and `"John"` is the value. The expression can be read "has name John", or "whose name is John". The above query can now be written:

```
a woman and parent : birth : (date : 1642 and place : in VA)
```

A semantics for our language, and a practical way to compute answers to queries in this language, is obtained by defining a translation to one-dimensional SPARQL queries. Graph patterns are given in the abstract syntax (constructs: AND, UNION, FILTER) defined in [PAG06], rather than the (equivalent) concrete SPARQL syntax, for the sake of simplicity and because it provides a necessary extension for translating negation (construct: MINUS). The empty graph pattern is denoted by 1.

Definition 4. *Let C be a complex class. The SPARQL translation of C is defined by*

$$\Gamma(C) = \texttt{SELECT ?}x\ \texttt{WHERE}\ f(g)$$

where $x \in V$ is a fresh variable not occurring in C, and $(g, f) = \gamma(x, C)$ (f is a function). The table below defines γ by induction on complex classes and complex properties. $\gamma(x, C)$ returns a graph pattern g, and a graph pattern modifier f, that together represent the fact that x is an instance of the complex class C. $\gamma(x, P^\alpha, y)$ is a graph pattern representing the complex property P between x and y, under the relation closure α. α is a subset of $\{?, +\}$, where ? (resp. +) denotes the reflexive (resp. transitive) closure of a binary relation. For every $i \in \mathbb{N}$, we assume $(g_i, f_i) = \gamma(x, C_i)$.

expression	graph pattern	graph pattern modifier
r	1	$\lambda g.(g$ `FILTER ?`$x = r)$
`?`v	1	$\lambda g.(g$ `FILTER ?`$x = ?v)$
`?`	1	$\lambda g.g$
`a` u	$(?x, \texttt{rdf:type}, u)$	$\lambda g.g$
$P\ C$	g_1 `AND` g_2	f_2
	where y is a fresh variable, $g_1 = \gamma(x, P^{\emptyset}, y)$, $(g_2, f_2) = \gamma(y, C)$	
C_1 `and` C_2	g_1 `AND` g_2	$\lambda g.(f_2(f_1(g)))$
`not` C_1	1	$\lambda g.(g$ `MINUS` $f_1(g_1))$
C_1 `or` C_2	1	$\lambda g.(g$ `AND` $(f_1(g_1)$ `UNION` $f_2(g_2)))$
p `:`	$(?x, p^{\alpha}, ?y)$	
p `of`	$(?y, p^{\alpha}, ?x)$	
p `with`	$(?x, p^{\alpha}, ?y)$ `UNION` $(?y, p^{\alpha}, ?x)$	
`opt` P	$\gamma(x, P^{\{?\}\cup\alpha}, y)$	
`trans` P	$\gamma(x, P^{\{+\}\cup\alpha}, y)$	

Compared to SPARQL, our language is restricted to one-dimensional relations, which makes the SPARQL contruct OPT irrelevant. This restriction is balanced to some extent by navigation (see end of Section 4.2). SPARQL allows for variables in predicate position, which is not directly possible in our language, but indirecly possible through the reification of triples. However, our language has native general negation, and reflexive/transitive closure. Perez et al. have shown that general negation is expressible in SPARQL, but in a very cumbersome way [PAG06].

We can now define the second part of a local view, the extent. It is simply defined as the answers to the SPARQL translation of the query.

Definition 5 (extent). *Let C be a complex class. The* extent *of C, noted $ext(C)$, is the set of resources that are answers to its SPARQL translation $\Gamma(C)$. Every element of the extent is called an* instance *of the complex class C.*

The extent of a query determines the focus concept the query leads to. The definition and the computation of the intent of this concept is not necessary in our framework. The intensional part of the local view is played by the *index*.

4.2 Summarization Index

The third part of the local view is the *index* which serves as a summary of the extent. Every *index term* is a descriptor of some or all resources in the extent. Therefore, every index term can be seen either as part of the intent of the focus concept, when shared by *all* instances; or as a refinement of the query, when shared by *some* of the instances.

Definition 6 (index term and intent term). *Let q be a complex class representing the query of a local view, and C be a complex class that contains only variables that also occur in q. C is an* index term *of q, which we note $C \in index(q)$,*

if $ext(q$ `and` $C) \neq \emptyset$. C *is an* intent term *of* q*, which we note* $C \in int(q)$*, if* $ext(q$ `and not` $C) = \emptyset$.

The number of index terms can be infinite, but in practice only a limited subset is presented to the user at any given time. Initially, a small index is presented, and then the user can expand it in a controlled way to see more index terms.

Instead of presenting the index terms as a flat list, they can be organized into a partial ordering $\leq$ that reflects subsumption relationships between them. Figure 1 shows how this partial ordering can be rendered as trees of complex classes. The number at the left of each index term is its count. Figure 1 gives on the right side the number of male Washington born in each place. This partial ordering needs not be complete w.r.t. subsumption because it does not affect query answering. The guiding criteria to design this partial ordering is that it should: be intuitive to users (i.e., they can anticipate the infered subsumptions), provide enough structure to the index, and be of practical complexity.

In the illustrations of this paper, we use RDFS inference through the properties `rdfs:subClassOf` and `rdfs:subPropertyOf`. In the partially ordered index, every class is placed under its superclasses. For instance, `a woman` $\leq$ `a person` $\leq$ `a thing`. Every property is placed under its superproperties, and also under its different closures. For instance, `father of` $\leq$ `parent of` $\leq$ `trans parent of`. From this ordering of complex properties, restrictions can also be ordered: $P_1\ C_1 \leq P_2\ C_2 \iff P_1 \leq P_2 \wedge C_1 \leq C_2$. For instance, `father of a man` $\leq$ `parent of a person`. Every index term is placed under the anonymous variable `?`, which then plays the role of the root of the whole index. Similarly, all restrictions in the form $P\ C$ are grouped under P `?`.

We have noted above that a reflexive and transitive complex property, i.e., in the form of `opt trans` P, is a partial ordering. This partial ordering can be used to organize the index because the following subsumption holds for every complex property P, and every resources r_1, r_2: `opt trans` $P\ r_1 \leq$ `opt trans` $P\ r_2 \iff r_1 \in ext($`opt trans` $P\ r_2)$. This is illustrated in Figure 1, on the right side, by the birthplace of male Washington's (recall that `in` = `opt trans part of`): e.g., `birth : place : in Westmorland` $\leq$ `birth : place : in VA`, because `Westmorland` $\in ext($`in VA`$)$ ("Westmorland is in VA"). Therefore, the tree under the index term `birth : place : in ?` forms a taxonomy of locations, even if each location is represented in the RDF graph as a resource, and not as a class. Similarly, a descendancy chart of the ancestors of the selected people is obtained under the index term `opt trans parent : ?`, showing under each individual its children, and this recursively.

The index alleviates to some extent our restriction to one-dimensional queries. Assume the SPARQL query `SELECT ?x ?y WHERE { ?x rdf:type gen:man . ?x gen:mother ?y }`. By setting the query to `a man`, and by expanding the index term `mother : ?`, the index gives the list of mothers of a man, and for each mother, how many male children she has. A highlighting mechanism allows to select a man in the extent to discover who is his mother; and symmetrically, to select a mother in the index to discover which are her children. This presentation is also more compact than a listing of all pairs (man, mother).

5 User Interaction: Navigation Links

A local view is determined by its query. The query determines the extent and the index as presented above. By default, the user is initially presented with the local view of the most general query `?`, whose extent is the set of all resources defined in the dataset. In their search for information, users need to change the focus, i.e., to change the current query. In retrieval search, users are looking for a particular set of resources, and try and find the query whose extent matches this set of resources. For instance, to answer the question "Which women are married to a Washington?", we can use the query `a woman and married with lastname : Washington`. In exploratory search, users are looking for patterns in data rather than for particular resources. For instance, if users is interested in the birthplace of people having lastname "Washington", they can set the query to `lastname : Washington`, and then explore the index under the index term `birth : place : in ?`. They obtain a natural hierarchy of places, where each place is annotated by the proportion of the Washington's that were born in this place. The index also informs about their birthdates, ancestors, descendants, etc.

A well informed user can of course directly type in queries. However, as explained in the introduction, this requires not only to have good knowledge of the query language (syntax and semantics) and of the domain-specific vocabulary (e.g., `man`, `place`), but also of the contents of the dataset if one wants to avoid empty results. This is paradoxical as the less we know a dataset, the more we need to search in it. We propose to define *navigation graphs* whose nodes are queries, hence local views, and whose edges are *navigation links* that users can follow.

Definition 7 (navigation graph). *A* navigation link *is a triple* $(q\ l\ q')$*, where* q *is the* source query*,* q' *is the* target query*, and* l *is the* label *of the navigation link. A* navigation graph G *is a set of navigation links that is deterministic, i.e., if* $(q\ l\ q'_1)$ *and* $(q\ l\ q'_2)$ *are 2 navigation links of* G*, then* $q'_1 \equiv q'_2$ *(*$\equiv$ *denotes query equivalence).*

Definition 8 (local links). *Let* G *be a navigation graph, and* q *be a query. The* local links *of* q *in* G*, noted* $Links_G(q)$*, is the set of navigation links in* G *whose source query is* q*. A navigation graph* G *is* locally finite *iff* $Links_G(q)$ *is finite for every complex class* q*.*

In the following, we first define the different kinds of navigation links, i.e., the *navigation modes*. Then a navigation graph is proved consistent, i.e., never leads to empty results. Finally, a locally finite navigation graph is proved complete, e.g., can lead to arbitrary queries/local views. These two navigation graphs define bounds between which every navigation graph is both consistent and complete.

5.1 Navigation Modes

There are only three navigation modes: *zoom-in*, *naming*, and *reversal*. Each navigation mode determines the target query in function of the source query

and an additional argument. A zoom-in applies to a complex class, a naming applies to a variable name (generated or user-given), and a reversal applies to a part of the source query. If we see the query in its SPARQL form, the zoom-in extends the graph pattern, while the reversal changes the variable in the SELECT clause. The naming makes a variable of the SPARQL query visible in the LIS query.

Definition 9 (zoom-in). *Let q be a query, and C be a complex class. A triple* `(`q `[zoom-in` C`]` q'`)` *is a* zoom-in *navigation link iff* $q' =$ `(`q `and` C`)`.

Zoom-in is mostly useful when the extent of the resulting query is strictly smaller and not empty. This is obtained when using index terms that are *not* intent terms. This useful distinction can be made visible in the interface by different renderings (e.g., font-color), and annotations (e.g., count).

Naming works similarly to zoom-in, but applies to a fresh variable, i.e., not occurring in the initial query, while zoom-in is expected to apply to variables already occurring in the source query.

Definition 10 (naming). *Let q be a query, and* `?`v *be a variable. A triple* `(`q `[naming ?`v`]` q'`)` *is a* naming *navigation link iff the variable v does not occur in q, and* $q' =$ `(`q `and ?`v`)`.

Naming does not change the extent, because it produces a query that is equivalent to the initial query, but it introduces a new variable in the query, and hence in the index. Subsequent zoom-in navigation links on these variables allow to form cycles in the graph pattern of the query.

Reversal does not change the graph pattern, but it changes the variable that appear in the SELECT clause. Indeed, in a LIS query, the focus is only on one variable, and it is useful to change this focus. Therefore, a reversal changes the extent, and hence the focus. A difficulty is that not all variables in the SPARQL pattern appear in its corresponding LIS query. In a reversal navigation link, the new variable is implicitly designated by a part of the query: i.e., an occurrence of a complex class or complex property in the query. Reversal is undefined when the part of the query is in the scope of union or complement.

Definition 11 (reversal). *Let q be a query, and e be a part of q. A triple* `(`q `[reversal` e`]` q'`)` *is a* reversal *navigation link iff e does not occur in the scope of a* union *or* complement*, and* $q' = \rho(?, \underline{q})$*. The following table defines $\rho(q', \underline{C})$ by induction on the complex class C. The underlined part indicates in which part of the query the selected element e stands. The first parameter q' is used as an accumulator in the building of the target query.*

complex class C	*result of* $\rho(q', \underline{C})$
$\underline{P}\ C'$ *when* $\sigma(P) = subject$	$\rho(q', \underline{P\ C'})$
$\underline{P}\ C'$ *when* $\sigma(P) = object$	$\rho(q', P\ \underline{C'})$
$P\ \underline{C'}$	$\rho(P^{-1}\ q', C')$
$\underline{C_1}$ `and` C_2	$\rho(q'$ `and` $C_2, C_1)$
C_1 `and` $\underline{C_2}$	$\rho(q'$ `and` $C_1, C_2)$
otherwise	q' `and` C

This definition needs a definition for the inverse of a complex property (P^{-1}), and for what a complex property refers to, whether the subject or the object of the property ($\sigma(P)$). The following table provides these definitions by induction on complex properties:

complex property P	*inverse* P^{-1}	*reference* ($\sigma(P)$)
p `:`	p `of`	*object*
p `of`	p `:`	*subject*
p `with`	p `with`	*subject*
`opt` P	`opt` P^{-1}	$\sigma(P)$
`trans` P	`trans` P^{-1}	$\sigma(P)$

We illustrate reversal with two examples, starting with the query already presented earlier: q = `a woman and parent : birth : (date : 1642 and place : in VA)`.

- q `[reversal place :] place of (birth of parent of a woman and date : 1642) and in VA`
  new focus on "where in Virginia a parent of a woman was born in 1642"
- q `[reversal 1642] 1642 and date of (place : in VA and birth of parent of a woman)`
  new focus on "the date 1642 of the birth in VA of a parent of a woman"

We now define a generic navigation graph, parameterized by a *vocabulary* of complex classes.

Definition 12 ($\mathcal{C}$-navigation graph). *Let $\mathcal{C}$ be a set of complex classes, called* vocabulary. *The $\mathcal{C}$-navigation graph G defines for every source query q the set of local navigation links, $Links_G(q)$, as follows:*
- a zoom-in link for each $C \in index(q) \cap \mathcal{C}$;
- a zoom-in link for each `not` *C s.t. $C \in \mathcal{C} \setminus int(q)$;*
- a naming link for one fresh variable;
- and a reversal link for each part of q not occurring in a union or complement.

5.2 Navigation Consistency

We first define *consistency* for navigation links and navigation graphs.

Definition 13 (navigation consistency). *A navigation link* `(`q l q'`)` *is* consistent *w.r.t. a dataset iff it preserves the existence of answers, i.e., $ext(q) \neq \emptyset$ implies $ext(q') \neq \emptyset$. A navigation graph is consistent iff its links are all consistent.*

A zoom-in link is consistent if it applies to an index term or to the complement of non-intent term of the query, so that every index term represents one or two consistent navigation links. All naming and reversal links are consistent.

Lemma 1. *Let q be a query. For every complex class $C \in index(q)$, the navigation link* `(`q `[zoom-in` C`]` q'`)` *is consistent; and for every complex class $C \notin int(q)$, the navigation link* `(`q `[zoom-in not` C`]` q'`)` *is consistent.*

Proof. By definition of $index(q)$, $int(q)$, and navigation link consistency. □

Lemma 2. *Let q be a query, and v be a variable not occurring in q. The navigation link* (`q [naming ?v] q'`) *is consistent.*

Proof. As `?v` does not occur in q, $q' =$ `q and ?v` is equivalent to `q and ?`, which is equivalent to q. □

Lemma 3. *Let q be a query, and e be a part of the query q not occurring in the scope of a union or complement. The navigation link* (`q [reversal e] q'`) *is consistent.*

Proof. It can be proved that the source and target query of a reversal link define the same SPARQL query, up to the renaming of variables, and the possible replacement of the variable in the SELECT clause. If the source query q has answers, this implies that every variable in the conjunctive part of the graph pattern (not in the scope of an union or complement pattern) has substitution values. Therefore, the new variable in the SELECT clause has substitution values. Hence, the target query q' also has answers. □

Theorem 1. *For every vocabulary $\mathcal{C}$, the $\mathcal{C}$-navigation graph is consistent.*

Proof. This is a direct consequence of previous definitions and lemmas. □

This implies that, given that the initial query has answers, a user who only follows navigation links in the navigation graph will never fall in a dead-end, which is a frequent cause of frustration in information systems.

5.3 Navigation Completeness

A navigation graph is complete if every query is reachable from the query `?`.

Definition 14 (navigation completeness). *A navigation graph G is* complete *iff for every query q whose extent is not empty, there exists a finite sequence of navigation links* $(q_0\ l_1\ q_1\ \ldots q_{n-1}\ l_n\ q_n)$*, where $q_0 =$* `?`*, $q_n = q$, and for every $i \in [1, n]$, $(q_{i-1}\ l_i\ q_i)$ is a navigation link of G. We call such a sequence, a* navigation path.

For practical use, a navigation graph has to be locally finite, i.e., only a finite set of local navigation links are suggested in any local view. Under this constraint we define a vocabulary for which completeness is achieved for *conjunctive queries*, i.e., queries without unions and with complements restricted to terms in $\mathcal{C}$.

A number of semantic web query languages are equivalent to conjunctive queries, and provide neither negation nor disjunction (e.g., OWL-QL [FHH04]). In our approach, negation and disjunction can still be introduced at any step of a navigation by editing the query by hand.

Theorem 2 (locally-finite conjunctive-complete navigation graph). *Let $\mathcal{C}$ be a finite vocabulary of complex classes containing at least resources (URIs, literals), variables, and unqualified restrictions* `P ?`*. The $\mathcal{C}$-navigation graph G is locally-finite and complete for* conjunctive queries.

Proof. For every query q, the set of local navigation links $Links_G(q)$ is finite because in a given dataset (1) there is a finite number of URIs, and hence of properties, (2) only literals present in the extent belong to the index, and this extent is always finite, and (3) only variables occurring in q belong to the index.

For completeness, it suffices to prove that for every complex classes q_0, and C such that C is a conjunctive query and $ext(q_0 \texttt{ and } C) \neq \emptyset$, there exists a path in G from q_0 to $q_1 = q_0 \texttt{ and } C$. In particular, setting $q_0 = \texttt{?}$ and $C = q$, we obtain that there exists a path from `?` to `(? and` q`)`, which is equivalent to q. We proceed by induction on the complex class C:

$C = r$**:** there is a path `(`q_0 `[zoom-in` r`]` q_1`)` (because $r \in \mathcal{C}$)
$C =?v,\ ?v \in q_0$**:** there is a path `(`q_0 `[zoom-in ?`v`]` q_1`)` (because `?`$v \in \mathcal{C}$)
$C =?v,\ ?v \notin q_0$**:** there is a path `(`q_0 `[naming ?`v`]` q_1`)`
$C =$**?:** $q_1 = q_0$ `and ?` $\equiv q_0$
$C =$ `a` u**:** $C \equiv$ `rdf:type` u (Definition 4)
$C = P\ C'$**:** (1) there is a path `(`q_0 `[zoom-in` P `?]` q_0 `and` P `? [reversal ?]` P^{-1} q_0 `and ?` $\equiv$ P^{-1} q_0`)`, (2) there is path from P^{-1} q_0 to `(`P^{-1} q_0 `and` C'`)` by induction on C', (3) there is a path `(`P^{-1} q_0 `and` C' `[reversal` q_0`]` q_0 `and` P C $= q_1$`)`. Induction in (2) is justified because `(`P^{-1} q_0 `and` C'`)` is a reversal of q_1, and every reversal link is consistent.
$C = C_1$ `and` C_2**:** (1) $q_1 \equiv$ `(`q_0 `and` C_1`)` `and` C_2 (associativity), (2) there is a path from q_0 to $q_2 =$ `(`q_0 `and` C_1`)` by induction on C_1, (3) there is a path from q_2 to `(`q_2 `and` C_2`)` by induction on C_2. The induction in (2) is justified because q_2 is more general than q_1, and hence $ext(q_2) \neq \emptyset$.
$C =$ `not` C_1, $C_1 \in \mathcal{C}$**:** there is a path q_0 `[zoom-in not` C_1`]` q_1. □

In the index, the set of resources can be seen as the list of answers to the query, and can be presented page by page, like in web search engines. The set of properties is organized into a subsumption hierarchy that can be expanded on demand by users. Instead of showing up to 12 complex properties for every basic property p, only p `:` and p `of` can be displayed. Their various closures are accessible by toggling each closure on/off when applying zoom-in (see check-boxes in Figure 1). The vocabulary used to compute the index and local navigation links in our prototype is richer than the base vocabulary of Theorem 2, in order to provide richer summaries of query results. The index also contains the hierarchy of classes (`a` u), and the user can expand restrictions recursively, i.e., each P `?` is refined into P C, where C is derived in the same way the main index is derived from `?`. Zoom-in is performed by double-clicking an index term, naming is performed by pushing a button that generates a fresh variable, and reversal is performed by clicking on a content word of the query (e.g., resource, variable, class, property). The interface offers three short-hand navigation links on index terms (semi-colon is used to compose navigation links): `[home]` (resets the query to `?`), `[pivot` C`]` = `[home; zoom-in` C`]`, `[cross` P C`]` = `[zoom-in` P C`; reversal` C`]`.

Then, the above query `a woman and parent : birth : (date : 1642 and place : in VA)` is accessible from `?` through the following navigation path: `[zoom-in a woman; cross parent : ?; cross birth : ?; zoom-in`

`date : 1642; zoom-in place : in VA; reversal a woman]`. After selecting women, the user moves to their parents, and then to the birth of their parents. By expanding recursively index terms, the user discovers birthdates and birthplaces, and select a date (1642), and a place (in VA). Finally, the user comes back to the point of view of women, now restricted to those whose some parent was born in 1642 in Virginia.

A more complex query with a cycle and a restricted complement is: `a person and ?X and birth : date : ?D and mother : mother of (not ?X and birth : date : ?D)`, which retrieves people with a twin sibling. It can be reached through the navigation path: `[zoom-in a person; naming ?X; cross birth :; cross date :; naming ?D; reversal ?X; cross mother :; cross mother of; zoom-in not ?X; cross birth :; cross date :; zoom-in ?D; reversal ?X]`.

6 Conclusion

We have defined local views over RDF graphs that serve both for summarization and navigation. Each loval view provides a set of local links that users can follow to reach other local views. The navigation graph induced by local views is both consistent and conjunctive-complete. Compared to existing conceptual navigation systems, our query language adds variables, binary relations between objects, negation and disjunction, thus covering most of the expressivity of SPARQL. Compared to SW query languages, we provide the same benefits as faceted search, i.e., exploratory search, but a larger fragment of the query language is reachable by navigation. Furthermore, consistency and completeness of navigation are proved formally.

Preliminary results from a user evaluation shows that all subjects could answer simple questions, like "Which men were born in 1659?"; and at least half of the subjects could answer complex questions involving variables, negation or disjunction, like "Which women have a mother whose death's place is not Warner Hall?" or "Who was born the same year as his/her spouse?". The simple questions match the expressivity of other faceted search systems, while complex questions are beyond their scope.

References

[BCM+03] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)

[BD08] Baader, F., Distel, F.: A finite basis for the set of EL-implications holding in a finite model. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 46–61. Springer, Heidelberg (2008)

[CR96] Carpineto, C., Romano, G.: A lattice conceptual clustering system and its application to browsing retrieval. Machine Learning 24(2), 95–122 (1996)

[DE08] Ducrou, J., Eklund, P.: An intelligent user interface for browsing and search MPEG-7 images using concept lattices. Int. J. Foundations of Computer Science 19(2), 359–381 (2008)

[Fer09] Ferré, S.: Camelis: a logical information system to organize and browse a collection of documents. Int. J. General Systems 38(4) (2009)

[FHH04] Fikes, R., Hayes, P.J., Horrocks, I.: OWL-QL - a language for deductive query answering on the semantic web. J. Web Semantic 2(1), 19–29 (2004)

[FR04] Ferré, S., Ridoux, O.: An introduction to logical information systems. Information Processing & Management 40(3), 383–419 (2004)

[FRS05] Ferré, S., Ridoux, O., Sigonneau, B.: Arbitrary relations in formal concept analysis and logical information systems. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) ICCS 2005. LNCS (LNAI), vol. 3596, pp. 166–180. Springer, Heidelberg (2005)

[GW99] Ganter, B., Wille, R.: Formal Concept Analysis — Mathematical Foundations. Springer, Heidelberg (1999)

[HHNV07] Hacene, M.R., Huchard, M., Napoli, A., Valtchev, P.: A proposal for combining formal concept analysis and description logics for mining relational data. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 51–65. Springer, Heidelberg (2007)

[HKR09] Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC, Boca Raton (2009)

[HvOH06] Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous semantic web repositories. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 272–285. Springer, Heidelberg (2006)

[Mar06] Marchionini, G.: Exploratory search: from finding to understanding. Communications of the ACM 49(4), 41–46 (2006)

[ODD06] Oren, E., Delbru, R., Decker, S.: Extending faceted navigation to RDF data. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 559–572. Springer, Heidelberg (2006)

[PAG06] Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006)

[RKH07] Rudolph, S., Krötzsch, M., Hitzler, P.: Quo vadis, CS? - on the (non)-impact of conceptual structures on the semantic web. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 464–467. Springer, Heidelberg (2007)

[ST09] Sacco, G.M., Tzitzikas, Y. (eds.): Dynamic taxonomies and faceted search. The information retrieval series. Springer, Heidelberg (2009)

[VR08] Völker, J., Rudolph, S.: Lexico-logical acquisition of OWL-DL axioms. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 62–77. Springer, Heidelberg (2008)

An Approach to Exploring Description Logic Knowledge Bases

Felix Distel

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

Abstract. This paper is the successor to two previous papers published at the ICFCA conference. In the first paper we have shown that in the Description Logics $\mathcal{EL}$ and $\mathcal{EL}_{\text{gfp}}$, the set of general concept inclusions holding in a finite model always has a finite basis. An exploration formalism that can be used to obtain this basis was presented in the second paper. In this paper we show how this formalism can be modified such that counterexamples to GCIs can be provided in the form of ABox-individuals. In a second part of the paper we examine which description logics can be used for this ABox.

1 Introduction

Description Logics (DLs) are a formalism for representing knowledge that has gained international recognition during the last decade [3]. They play a significant role in the Semantic Web Community, in particular because of the OWL language which is essentially a variant of an expressive DL [10].

A DL knowledge base usually consists of two parts. The first part, the TBox is used to describe the terminology of the knowledge base. It contains general concept inclusion (GCIs), i. e. statements of the form $C \sqsubseteq D$. Here C and D are concept descriptions written using a set of so-called concept constructors, concept names and role names. Different DL languages use different concept constructors. However, all DL languages provide a formal, well-defined model based semantics for the concept descriptions. A model $i = (\Delta_i, \cdot^i)$ consists of a set Δ_i and a function $\cdot^i$ that maps concept descriptions C to subsets $C^i \subseteq \Delta_i$. The second part of the knowledge base is the ABox. It contains knowledge about individuals. One can for example assert that an individual `Henry` belongs to the concept `Father` or that there is a `hasChild` role leading from `Henry` to `Jane`.

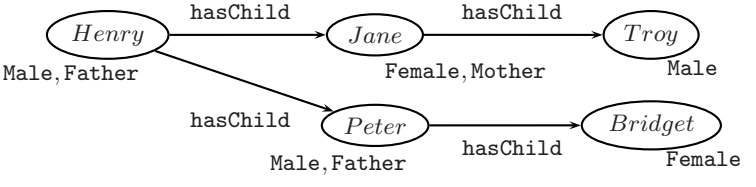


Fig. 1. A Model of a Family of Three Generations

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 209–224, 2010.

An important aspect are the open world semantics of ABoxes. If it is not stated that `Henry` is a `Father` then it is not assumed that `Henry` is not a `Father`.

Writing knowledge bases can be a difficult process, in particular because experts in the domain of the knowledge base are usually not experts in DL. In order to help them to find the right GCIs to add to their TBox, one approach is to use a formalism that is inspired by attribute exploration from Formal Concept Analysis (FCA) [9]. In the formalism that has been presented in a previous ICFCA Paper [6] it is assumed that the domain of the knowledge base can be represented as a DL model, and that this model is completely known to a human expert. In this formalism the expert does not have to come up with GCIs herself. Instead the system suggests GCIs that she can either add to the TBox, or reject by providing a counter-example. This approach is often referred to as *knowledge base completion*.

Let us assume that the domain was represented by the model of a family of three generations from Figure 1. The system might come up with a GCI like $\texttt{Father} \sqsubseteq \texttt{Male} \sqcap \exists \texttt{hasChild}.\top$, i. e. "Every father is male and has a child." The expert would obviously accept this GCI and add it to the knowledge base. If, however, the system comes up with the GCI $\texttt{Father} \sqsubseteq \texttt{Mother}$, i. e. "Every father is a mother", then the expert would reject it and add e. g. `Henry` as a counter-example.

The GCIs from the example are written in the lightweight description logic $\mathcal{EL}$. $\mathcal{EL}$ is less expressive than most other standard DLs but has the advantage that standard reasoning tasks are tractable [8]. This is one of the reasons why tractable extensions of $\mathcal{EL}$ are used for large scale biomedical ontologies such as SNOMED [12] and the Gene Ontology [13].

Our algorithm from the previous ICFCA paper also uses a tractable extension of $\mathcal{EL}$, $\mathcal{EL}_{\text{gfp}}$, which allows the algorithm to generate concept descriptions that are cyclic. The major weakness of our previous algorithm is the way in which counter-examples are provided. It uses connected submodels which use a closed-world semantics. The submodel is extended every time the expert provides a counter-example. Let us assume the expert wants to state that `Henry` is a counter-example to the GCI $\texttt{Father} \sqsubseteq \texttt{Mother}$. Assume that the expert only adds `Henry`, but not `Jane` or `Peter`, to the submodel and says that `Henry` is a `Father` but not a `Mother`. Because of the closed world semantics the algorithm would assume that `Henry` does not have children which would make `Henry` a counter-example to the GCI $\texttt{Father} \sqsubseteq \texttt{hasChild}.\top$. This is unwanted because $\texttt{Father} \sqsubseteq \texttt{hasChild}.\top$ does hold in the domain. The only way to avoid this effect is to add not only `Henry`, but also all of his direct or indirect role successors, in this case his children and grandchildren.

So the expert would need to add a lot more information than is actually needed to make `Henry` a counter-example without creating unwanted artefacts. This is inconvenient and can only be overcome by allowing open-world-semantics. In the DL-world the natural datastructure to keep track of individuals which provides an open-world semantics is an ABox. This paper will present an approach how to extend the algorithm from the previous paper to work with ABoxes as the

underlying datastructure. We will introduce minimal possible consequences as a central notion. Since this is ongoing work some important questions remain open, e. g. if and how minimal possible consequences can be computed effectively.

Due to space restrictions we cannot introduce Formal Concept Analysis. We assume that the reader is familiar with the basic notions from this field.

Related Work: There are two other works important works that try to combine FCA and DL. The work by Baader et al. provides a knowledge base completion formalism that also uses ABoxes as the underlying datastructure [7]. However, their algorithm does not perform knowledge base completion with respect to arbitrary GCIs written in a language like $\mathcal{EL}$. Instead they only allow conjunctions of previously defined concepts. The second approach by Rudolph can be used to compute a basis for the GCIs of a given DL model. The main difference compared to our approach lies in the way the GCIs are computed. While we construct a context on the fly, adding only a few interesting attributes at a time, Rudolph's approach successively increases role depth and adds all attributes up to a certain depth [11].

2 Preliminaries

The Description Logic $\mathcal{EL}$. Due to space restrictions we can only give a brief introduction to the DLs $\mathcal{EL}$ and $\mathcal{EL}_{\text{gfp}}$. $\mathcal{EL}$ concept descriptions are generated from a finite set $\mathcal{N}_C$ of concept names and a finite set $\mathcal{N}_r$ of role names as follows.

- concept names and the top concept $\top$ are $\mathcal{EL}$-concept descriptions;
- if C, D are $\mathcal{EL}$-concept descriptions and r is a role name, then $C \sqcap D$ and $\exists r.C$ are $\mathcal{EL}$-concept descriptions.

The tuple $\Sigma = (\mathcal{N}_C, \mathcal{N}_r)$ is called the *signature* of the concept description.

A $\mathcal{EL}$ model $i = (\Delta_i, \cdot^i)$ consists of a finite set Δ_i, the so-called domain of the model, and an interpretation function $\cdot^i$ mapping role names r to relations $r^i \subseteq \Delta_i \times \Delta_i$ and concept descriptions C to their extensions such that

$$\top^i = \Delta_i, \quad (C_1 \sqcap C_2)^i = C_1^i \cap C_2^i, \quad \text{and}$$
$$(\exists r.D)^i = \{d \in \Delta_i \mid \exists e \in D^i \text{ such that } (d, e) \in r^i\}.$$

Note that it suffices to define the interpretation function for role names and concept names. The interpretations of more complex concept descriptions can then be derived, recursively. Subsumption and equivalence between $\mathcal{EL}$-concept descriptions is defined in the usual way, i.e., C is subsumed by D (written $C \sqsubseteq D$) iff $C^i \subseteq D^i$ for all models i, and C is equivalent to D (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$.

TBoxes and ABoxes. A GCI is a statement of the form $C \sqsubseteq D$, where C and D are concept descriptions. We say that a GCI $C \sqsubseteq D$ *holds in a model* i if $C^i \subseteq D^i$ holds. Note that this is not the same as subsumption. An equivalence

statement is a statement of the form $A \equiv D$, where A is a concept name and D a concept description. $A \equiv D$ is said to *hold in* i if $A^i = D^i$.

TBoxes are sets of equivalence statements and GCIs. They fall into three categories.

- *Acyclic TBoxes* contain only equivalence statements where the left-hand side is not used in the concept description on the right-hand side implicitly or explicitly.
- *Cyclic TBoxes* contain only equivalence statements
- *General TBoxes* contain arbitrary GCIs.

A model i is said to be a model of a TBox $\mathcal{T}$ if all statements from $\mathcal{T}$ hold in i. In the case of cyclic TBoxes there exists also the notion of *greatest-fixpoint-models*. Informally, a model i is a greatest-fixpoint model of $\mathcal{T}$ if the interpretations of all concept names in i are maximal among all other models of $\mathcal{T}$ with the same domain. A more formal definition can be found in [2].

An *ABox* $\mathcal{A}$ is a set of concept assertions and role assertions, where a *role assertion* is of the form $r(a, b)$ and a *concept assertion* is of the form $A(a)$, with r a role name, A a concept name, and a and b so-called individual names. A *model* $i = (\Delta_i, \cdot^i)$ *of an ABox* $\mathcal{A}$ is a model where $\cdot^i$ is extended to map individual names a to individuals $a^i \in \Delta_i$ such that $a \in A^i$ for all concept assertions $A(a) \in \mathcal{A}$ and $(a, b) \in r^i$ for all role assertions $r(a, b) \in \mathcal{A}$.

The Description Logic $\mathcal{EL}_{\text{gfp}}$. $\mathcal{EL}_{\text{gfp}}$ is the extension of $\mathcal{EL}$ by cyclic concept definitions interpreted with greatest fixpoint (gfp) semantics. In $\mathcal{EL}_{\text{gfp}}$, we assume that the set of concept names is partitioned into the set $\mathcal{N}_{\text{prim}}$ of primitive concepts and the set $\mathcal{N}_{\text{def}}$ of defined concepts. We only allow concept definitions of the form

$$B_0 \equiv P_1 \sqcap \ldots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \ldots \sqcap \exists r_n.B_n \tag{1}$$

where $B_0, B_1, \ldots, B_n \in \mathcal{N}_{\text{def}}$, $P_1, \ldots, P_m \in \mathcal{N}_{\text{prim}}$, and $r_1, \ldots, r_n \in \mathcal{N}_r$. The empty conjunction (i.e., $m = 0 = n$) stands for $\top$.

Definition 1 ($\mathcal{EL}_{\text{gfp}}$-concept description). *A* $\mathcal{EL}_{\text{gfp}}$-concept description *is a tuple* $(A, \mathcal{T})$ *where* $\mathcal{T}$ *is a TBox and* A *is a defined concept occurring on the left-hand side of a definition in* $\mathcal{T}$*.*

Let $i = (\Delta_i, \cdot^i)$ be a model. The *extension* $(A, \mathcal{T})^i$ *of* $(A, \mathcal{T})$ in i is the set assigned to A by the gfp-model of $\mathcal{T}$ based on i. Subsumption and equivalence between $\mathcal{EL}_{\text{gfp}}$-concept descriptions is defined as in the case of $\mathcal{EL}$-concept descriptions. It is easy to see that acyclic $\mathcal{EL}_{\text{gfp}}$-concept descriptions (i.e., ones where the TBox component is acyclic) correspond exactly to $\mathcal{EL}$-concept descriptions.

It is difficult to obtain a good intuition about greatest-fixpoint semantics. Fortunately, there is an alternative characterization. Given a model i and an individual $x \in \Delta_i$ we can define the set of concept names assigned to x as $\text{names}_i(x) = \{A \in \mathcal{N}_{\text{prim}} \mid x \in A^i\}$. We denote the set of all r-successors of x in i by $xr^i = \{y \in \Delta_i \mid (x, y) \in r^i\}$.

For TBoxes that contain only concept definitions of the form (1) we introduce notations similar to those for models. For a defined concept B we denote by $\text{names}_{\mathcal{T}}(B)$ the set of all primitive concept names $P_1, \ldots, P_k$ that occur in the definition of B in $\mathcal{T}$. For a defined concept B_1 and a role name r we denote by $B_1 r_{\mathcal{T}}$ the set of all defined concept names B_2 for which the term $\exists r.B_2$ occurs in the definition of B_1 in $\mathcal{T}$. A simulation from a normalized TBox $\mathcal{T}$ to a model i is a relation $\zeta \subseteq \mathcal{N}_{\text{def}} \times \Delta_i$ where

(S1) $\text{names}_{\mathcal{T}}(B) \subseteq \text{names}_i(x)$ for all pairs $(B, x) \in \zeta$. , and
(S2) for all role names $r \in \mathcal{N}_r$, all pairs $(B, x) \in \zeta$ and all $E \in Br_{\mathcal{T}}$ there is some $y \in x r^i$ such that $(E, y) \in \zeta$ holds.

The following theorem enables us to check instance without using greatest fixpoints explicitly [1].

Lemma 1. *Let $C = (A_C, \mathcal{T}_C)$ be an $\mathcal{EL}_{\text{gfp}}$-concept description. Let $i = (\Delta_i, \cdot^i)$ be a model and $x \in \Delta_i$ an individual. Then it holds that $x \in C^i$ iff there is a simulation ζ from $\mathcal{T}_C$ to i that contains (A_C, x).*

Given a set of GCIs $\mathcal{B}$, we say that the GCI $C \sqsubseteq D$ follows from $\mathcal{B}$ if $C \sqsubseteq D$ holds in all models in which all GCIs from $\mathcal{B}$ hold. We say that $\mathcal{B}$ is a *basis* for the $\mathcal{EL}_{\text{gfp}}$-GCIs holding in i if $\mathcal{B}$ is

- *sound* for i, i. e. it contains only GCI that hold in i, and
- *complete* for i, i. e. any $\mathcal{EL}_{\text{gfp}}$-GCI holding in i follows from $\mathcal{B}$.

3 Results from Previous Work

Exploration as a method for knowledge base completion relies on the existence of an expert with complete knowledge about the domain of the knowledge base. For practical purposes we assume that the domain of the knowledge base ("the real world") can be represented as a model i of the final (complete) knowledge base. In this and the previous work i is called the *background model*. The goal of an exploration is to find a basis for the set of GCIs holding in i.

In doing this we face two major challenges: First, for most DLs it is not trivial to find a basis, even when the background model is known. Second, since the complete background model is unknown to the algorithms, the algorithm must gradually gain information about the background model by querying the expert. The first challenge has been adressed in [5] while a solution for the second problem is proposed in [6]. The purpose of this section is to recapitulate important notions from these two publications.

3.1 Model-Based Most Specific Concepts

Suppose we want to compute a basis for the GCIs holding in the model i from Figure 1. Suppose furthermore that we have decided (for example by using the algorithm described in [6]) that the $\mathcal{EL}_{\text{gfp}}$-concept description `Father` is an interesting premise for a GCI. We might add any of the GCIs $\texttt{Father} \sqsubseteq \texttt{Male}$,

or $\texttt{Father} \sqsubseteq \texttt{Male} \sqcap \exists \texttt{hasChild}.\top$ to the basis. However, if we decide to add the first one and later find out that we need to add also the second to ensure completeness we obtain redundance (because the first GCI follows from the latter). In an exploration setting this would mean that we would ask two questions where one would be enough. To avoid redundant questions, the idea is to be as specific as possible when choosing the right-hand side of a GCI. For the description logic $\mathcal{EL}_{\text{gfp}}$ model-based most specific concepts are what we need to find these conclusions.

Definition 2 (Model-Based Most Specific Concept). *Let $i = (\Delta_i, \cdot^i)$ be a finite model and $X \subseteq \Delta_i$ a set. The $\mathcal{EL}_{\text{gfp}}$-concept description C is the* most specific $\mathcal{EL}_{\text{gfp}}$-concept of X in i *if it is the least $\mathcal{EL}_{\text{gfp}}$-concept description such that $X \subseteq C^i$. By* least $\mathcal{EL}_{\text{gfp}}$-concept description *we mean that every other $\mathcal{EL}_{\text{gfp}}$-concept description $\bar{C}$ satisfying $X \subseteq \bar{C}^i$ also satisfies $C \sqsubseteq \bar{C}$.*

It is justified to speak of *the* model-based most specific concept (mmsc) because the model-based most specific concept is unique up to equivalence. We use the notation X^i to denote the mmsc of X. Mmsc for the description logic $\mathcal{EL}_{\text{gfp}}$ exist for all models $i = (\Delta_i, \cdot^i)$ and all sets $X \subseteq \Delta_i$ and can be computed effectively [5].

Lemma 2. *Let i be a model, $X, Y \in \Delta_i$ sets of objects and let C, D be $\mathcal{EL}_{\text{gfp}}$-concept descriptions. Then the following statements hold*

1. $X \subseteq Y \Rightarrow X^i \sqsubseteq Y^i$
2. $C \sqsubseteq D \Rightarrow C^i \subseteq D^i$
3. $X \subseteq X^{ii}$
4. $C^{ii} \sqsubseteq C$
5. $X^i \equiv X^{iii}$
6. $C^i = C^{iii}$
7. $X \subseteq C^i \Leftrightarrow X^i \sqsubseteq C$.

This lemma from [5] shows that GCIs of the form $C \sqsubseteq C^{ii}$ play a special rôle.

Lemma 3. *Let C, D be $\mathcal{EL}_{\text{gfp}}$-concept descriptions and i a finite $\mathcal{EL}_{\text{gfp}}$-model. Then $C \sqsubseteq C^{ii}$ holds in i. If $C \sqsubseteq D$ holds in i, then $C \sqsubseteq D$ follows from $\{C \sqsubseteq C^{ii}\}$.*

We have seen that mmsc can help to reduce redundancy. They are therefore useful when it comes to constructing finite sets of axioms for a given model.

3.2 An Algorithm for Axiomatizing a Given Model

In [6] an algorithm has been presented that can be used to axiomatize a given (known) model i (Algorithm 1). Given a finite model i as input Algorithm 1 will always terminate. Upon termination it will have produced a set Π_n of so-called premises P_k such that

$$\mathcal{B}_n := \{\textstyle\bigsqcap P_k \sqsubseteq (\bigsqcap P_k)^{ii} \mid P_k \in \Pi_n\}$$

is a basis for the GCIs holding in i.

The algorithm uses the notation $\bigsqcap U$, where U is a set of concept descriptions, to denote the concept $\bigsqcap U := \bigsqcap_{D \in U} D$.

Algorithm 1. Computing a basis for an a priori given model i

Input: finite model $i = (\Delta_i, \cdot^i)$
$M_0 := \mathcal{N}_C$, $\mathcal{S}_0 := \emptyset$
$\Pi_0 := \emptyset$, $P_0 := \emptyset$, $k := 0$
while $P_k \neq \mathsf{null}$ **do**
  $\Pi_{k+1} := \Pi_k \cup \{P_k\}$
  $M_{k+1} := M_k \cup \{\exists r.(\bigsqcap P_k)^{ii} \mid r \in \mathcal{N}_r\}$
  $\mathcal{S}_{k+1} := \{\{C\} \to \{D\} \mid C, D \in M_{k+1}, C \sqsubseteq D\}$
  $k := k + 1$
  if $M_k = M_{k-1} = P_k$ **then**
    $P_k := \mathsf{null}$
  else
    $P_k :=$ lectically next set of attributes that respects all implications in $\{P_j \to P_j^{''_k} \mid 1 \leq j < k\}$ and $\mathcal{S}_k$
  end if
end while

It uses some elements of FCA, in particular the next-closure algorithm. The connection between FCA and DL is made by so-called induced contexts. What we call induced contexts in this work are formal contexts whose attributes are concept descriptions and whose set of objects is the domain Δ_i of a finite model i. More formally, let i be a finite $\mathcal{EL}_{\text{gfp}}$-model and M a finite set of $\mathcal{EL}_{\text{gfp}}$-concept descriptions. The *context induced by M and i* is the formal context $\mathbb{K} = (G, M, I)$, where $G = \Delta_i$ and $I = \{(x, C) \mid C \in M \text{ and } x \in C^i\}$.

There are infinitely many possible concept descriptions and thus infinitely many possible attributes for an induced context. The most important idea in the construction of Algorithm 1 was that the set of attributes was not fixed in the beginning. Instead a new set of attributes M_k is generated during each iteration. The notation $\cdot^{''_k}$ denotes the $\cdot''$-operator from FCA computed in the context $\mathbb{K}_k$, where $\mathbb{K}_k$ denotes the context induced by M_k and i.

3.3 Exploration Using Submodels

Of the two main challenges that we have identified, the second one was constructing a set of axioms in a situation where the background model is not known to the algorithm. The only way to gain information about the model is to ask the expert. In [6] an algorithm has been presented that uses the familiar exploration principle. It generates a GCI and asks the expert whether this GCI holds in the background model. If so, the GCI is added to the set of axioms. Otherwise the expert is asked to provide a counterexample. Now the question is in what form these counterexamples should be provided. In [6] the counterexamples are provided in the form of connected submodels of the background model.

Thereby a submodel j of a model i is a model such that $\Delta_j \subseteq \Delta_i$ and $C^j = C^i \cap \Delta_j$ for all concept names C and $r^j = r^i \cup (\Delta_j \times \Delta_j)$ for all role names r. j is called a *connected submodel* if and only if for every $x \in \Delta_i$ and all $r \in \mathcal{N}_r$ if $x \in \Delta_j$ then all r-successors of x are also in Δ_j. Whenever a GCI is refuted

the expert is asked to provide a new model i_j that we call the *working model*. It is required to extend the previous working model i_{j-1}, to be a connected submodel of i and to contain a counterexample. Similar to Algorithm 1 it has been shown that this algorithm always terminates and produces a basis for the set of implications holding in i.

4 Replacing Models by ABoxes

4.1 Possible Consequences

We consider a setting where (instead of a connected submodel of the background model i) the expert provides a knowledge base consisting of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. For now, the background model i should be a model of the ABox $\mathcal{A}$ that contains the counterexamples and the TBox $\mathcal{T}$. Given $\mathcal{A}$ and $\mathcal{T}$ what can be said about the GCIs that hold in i? First, there are the GCIs that hold in *every* model of $\mathcal{A}$ and $\mathcal{T}$. These are the GCIs which are already known to hold in i. Therefore they are not interesting for a completion formalism.

On the other hand, there are the GCIs that hold in at least one model of $\mathcal{A}$ and $\mathcal{T}$. Since the background model i is unknown, it is possible that i is one of these models in which the GCI holds. So these GCIs are the ones we are interested in. Provided an $\mathcal{EL}_{\text{gfp}}$-concept description C we define the set of concept descriptions D that are *possible consequences* of C to be

$$\mathfrak{pc}_{\mathcal{A},\mathcal{T}}(C) = \{D \mid \exists j \text{ model of } \mathcal{A} \text{ and } \mathcal{T} : C^j \subseteq D^j\}.$$

Notice, that we do not make any requirements with respect to the language of the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$, except that they have a model-theoretic semantics with models as defined in Section 2. It may be different from $\mathcal{EL}_{\text{gfp}}$. The certain and possible consequences, however, are expressed in $\mathcal{EL}_{\text{gfp}}$.

Now, suppose we want to present to the expert a GCI $C \sqsubseteq D$ whose left-hand side is C. It does not make sense to ask this question, unless D is a possible consequence of C. Otherwise the answer would certainly be "No". So we have to choose D among the possible consequences of C.

Once the expert accepts a GCI, the algorithm should not have to generate another GCI with the same premise. This is why we introduce the notion of minimal possible consequences. D is said to be a minimal possible consequence of C if $D \in \mathfrak{pc}_{\mathcal{A},\mathcal{T}}(C)$ and D is minimal in $\mathfrak{pc}_{\mathcal{A},\mathcal{T}}(C)$ with respect to $\sqsubseteq$. The set of all minimal possible consequences of C is denoted by $\mathfrak{mpc}_{\mathcal{A},\mathcal{T}}(C)$. Unlike mmsc minimal possible consequences need not be unique up to equivalence. We will mostly be interested in GCIs over a fixed signature Σ. We introduce the notation $\mathfrak{pc}^{\Sigma}_{\mathcal{A},\mathcal{T}}(C)$ for the set of all possible consequences that are expressed using only the signature Σ. Analogously, we define $\mathfrak{mpc}^{\Sigma}_{\mathcal{A},\mathcal{T}}(C)$.

Those who are familiar with [7] will find that the $\mathcal{K}(\cdot)$-operator computes minimal possible consequences for the special case of a logic that allows only for conjunction.

4.2 Adapting the Exploration Algorithm

It is not yet known if (or rather for which logics) minimal possible consequences exist. This is work in progress and will not be considered here. For now, we assume that the knowledge bases considered here are written in a logic for which the existence of minimal possible consequences is guaranteed. We also assume that there exists an oracle to compute a minimal possible consequence for a given $\mathcal{EL}_{\text{gfp}}$-concept description C.

We show that under these assumptions Algorithm 1 requires only subtle modifications in order to function with ABoxes as underlying datastructure. The modified algorithm is presented as Algorithm 2. We assume that there is a background model i which is known to the expert. The input consists of a TBox $\mathcal{T}_0$ and an ABox $\mathcal{A}_0$ (instead of a model). We require that i is a model of the $\mathcal{T}_0$ and $\mathcal{A}_0$. The signature of the initial knowledge base is denoted by Σ_0.

The modification with respect to Algorithm 1 primarily consists in the addition of a second while-loop. Informally, the purpose of this inner while-loop is to find the proper conclusion D_k to a given premise $\bigsqcap P_k$. Since i is not explicitly given it is not possible to directly compute $(\bigsqcap P_k)^{ii}$ like in Algorithm 1.

Before we start to prove completeness, let us first clarify a few details about Algorithm 2. First of all, note that while the newly acquired GCIs (i. e. the P_k found in the algorithm) are formulated in $\mathcal{EL}_{\text{gfp}}$ we do not specify the logic of the underlying ABox and TBox. Using two different languages may seem unnatural at first, but is, unfortunately, necessary. This will become clear in Section 5.

In Line 19 $\text{pr}_M(C)$ denotes the *projection* of a concept description C to a set of concept descriptions M, i. e. the set $\text{pr}_M(C) = \{D \in M \mid C \sqsubseteq D\}$. The following lemma about projections in induced contexts has been proved in [4].

Lemma 4. *Let $U \subseteq M$ be any set of attributes in a context $\mathbb{K}$ induced by i and M. Then $U'' = \text{pr}_M\left((\bigsqcap U)^{ii}\right)$.*

A last remark concerns the changing signatures. In Line 8 the expert is asked to provide a new TBox $\mathcal{T}_j$ and ABox $\mathcal{A}_j$. We allow that new concept names that are not present in Σ_0 are used in $\mathcal{T}_j$ and $\mathcal{A}_j$. The motive behind this is that in certain logics new concept names are necessary to express that an individual is a counterexample to a GCI (cf. Section 5.2). Allowing new concept names yields one problem: It is not clear how to interpret the new concept names in the background model. In other words i is not a model of $\mathcal{T}_j$ and $\mathcal{A}_j$. That is why we introduce the notion of a *representation* of a model. $\mathcal{T}_j$ and $\mathcal{A}_j$ are called a representation of i if there is a model ι of $\mathcal{T}_j$ and $\mathcal{A}_j$ such that

- $\Delta_i = \Delta_\iota$, and
- for all $\mathcal{EL}_{\text{gfp}}$-concept descriptions over the smaller signature Σ_0 it holds that $C^i = C^\iota$.

Once the expert has accepted a GCI, the algorithm should not need to consider the same premise P_k again. Lemma 5 shows why this is indeed the case.

Lemma 5. *Whenever Algorithm 2 leaves the inner while-loop (Lines 6 to 10) it holds that $D \equiv (\bigsqcap P_k)^{ii}$.*

Algorithm 2. The ABox Exploration Algorithm

Input: ABox $\mathcal{A}_0$, TBox $\mathcal{T}_0$ with signature Σ_0
$M_0 := \mathcal{N}_{\text{prim}}$, $\mathcal{S}_0 := \emptyset$
$\mathit{\Pi}_0 := \emptyset$, $P_0 := \emptyset$, $k := 0$, $j := 0$
while $P_k \neq \mathsf{null}$ **do**
  Obtain $D \in \mathfrak{mpc}^{\Sigma_0}_{\mathcal{T}_j,\mathcal{A}_j}(\bigsqcap P_k)$ from oracle
  while expert refutes $\bigsqcap P_k \sqsubseteq D$ **do**
    $j := j + 1$
    Ask the expert for a new knowledge base $(\mathcal{T}_j, \mathcal{A}_j)$ that extends $(\mathcal{T}_{j-1}, \mathcal{A}_{j-1})$, and is a representation of i.
    Obtain $D \in \mathfrak{mpc}^{\Sigma_0}_{\mathcal{T}_j,\mathcal{A}_j}(\bigsqcap P_k)$ from oracle
  end while
  $D_k = D$
  $\mathit{\Pi}_{k+1} := \mathit{\Pi}_k \cup \{P_k\}$
  $M_{k+1} := M_k \cup \{\exists r.D_k \mid r \in \mathcal{N}_r\}$
  $\mathcal{S}_{k+1} := \{\{C\} \to \{E\} \mid C, E \in M_{k+1}, C \sqsubseteq E\}$
  $k := k + 1$
  if $M_k = M_{k-1} = P_k$ **then**
    $P_k := \mathsf{null}$
  else
    $P_k :=$ lectically next set of attributes that respects all implications in $\{P_l \to \mathrm{pr}_{M_k}(D_l) \mid 1 \leq l < k\}$ and $\mathcal{S}_k$
  end if
end while

Proof. The algorithm will only leave the inner while-loop when the expert states that $\bigsqcap P_k \sqsubseteq D$ holds in i. This means that $(\bigsqcap P_k)^i \subseteq D^i$ is true. Lemma 2 implies that $(\bigsqcap P_k)^{ii} \sqsubseteq D$. Because $\mathcal{A}_j$ and $\mathcal{T}_j$ are a representation, there must be a model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ such that for all $\mathcal{EL}_{\text{gfp}}$-concept descriptions C over the smaller signature Σ_0 it holds that $C^i = C^\iota$. Since $\bigsqcap P_k$ and $(\bigsqcap P_k)^{ii}$ use only the signature Σ_0 it follows that $(\bigsqcap P_k)^\iota = (\bigsqcap P_k)^i \subseteq (\bigsqcap P_k)^{iii} = ((\bigsqcap P_k)^{ii})^\iota$. This shows that $(\bigsqcap P_k)^{ii}$ is a possible consequence of $\bigsqcap P_k$. Since D is minimal among the possible consequences of $\bigsqcap P_k$ we obtain $D \sqsubseteq (\bigsqcap P_k)^{ii}$. Thus $D \equiv (\bigsqcap P_k)^{ii}$.

Theorem 1 (Completeness). *Assume that Algorithm 2 terminates after the n-th iteration of the outer while loop. Then the set of GCIs $\mathcal{B} = \{\bigsqcap P_k \sqsubseteq D_k \mid 0 \leq k \leq n\}$ is complete for the background model i.*

Proof. We prove completeness by showing that Algorithm 2 finds exactly the same GCIs as Algorithm 1 initialised with the full background model i. This is done by induction. Let P_k, $\mathit{\Pi}_k$, M_k and S_k represent the outputs of Algorithm 1. Let $\mathbf{P}_k$, $\mathbf{\Pi}_k$, $\mathbf{M}_k$ and $\mathbf{S}_k$ represent the respective outputs of Algorithm 2. We prove by induction over k that

$$P_k = \mathbf{P}_k, \qquad \mathit{\Pi}_k = \mathbf{\Pi}_k, \qquad M_k = \mathbf{M}_k, \qquad S_k = \mathbf{S}_k \tag{2}$$

Base Case: The case $k = 0$ is trivial. *Step Case:* Assume that (2) holds for all $k < k_0$. *Part 1.* $\mathit{\Pi}_{k_0} = \mathbf{\Pi}_{k_0}$, $M_{k_0} = \mathbf{M}_{k_0}$, $S_{k_0} = \mathbf{S}_{k_0}$ follow immediately

from the induction hypothesis and Lines 5-7 in Algorithm 1 and Lines 12-14 in Algorithm 2. *Part 2.* We show that $P_{k_0} = \mathbf{P}_{k_0}$. To do this, we only need to show that $\mathrm{pr}_{\mathbf{M}_{k_0}}(\mathbf{D}_l) = P_l^{''k_0}$ for all $1 \leq l < k_0$ (see Line 12 of Algorithm 1 and Line 19 of Algorithm 2). Lemma 4 shows that $P_l^{''k_0} = \mathrm{pr}_{M_{k_0}}\left((\bigsqcap P_l)^{ii}\right)$. By induction hypothesis and Part 1 we obtain $P_l^{''k_0} = \mathrm{pr}_{\mathbf{M}_{k_0}}\left((\bigsqcap \mathbf{P}_l)^{ii}\right)$. Then Lemma 5 proves that $\mathrm{pr}_{\mathbf{M}_{k_0}}(\mathbf{D}_l) = P_l^{''k_0}$ for all $1 \leq l < k_0$, and therefore $P_{k_0} = \mathbf{P_{k_0}}$.

This finishes the induction proof. So we have shown that $\mathbf{P}_k = P_k$ for all $k \in \{1, \ldots, n\}$. Lemma 5 proves $\mathbf{D}_k = (\bigsqcap \mathbf{P}_k)^{ii} = (\bigsqcap P_k)^{ii}$. Hence the set of GCIs $\mathcal{B}$ that is found by Algorithm 2 is exactly the same as the set $\mathcal{B}_n$ that Algorithm 1 computes with the full background model i as input. Since Algorithm 1 is complete, Algorithm 2 must also be complete.

Termination, however, is more difficult. If the algorithm does not get stuck in the inner while-loop (Lines 6 to 10) then it is guaranteed to terminate. This is because outside the inner while loop it behaves just like Algorithm 1, and Algorithm 1 terminates. In summary, there remain two issues to be adressed: The existence and computation of minimal possible consequences and termination of the above algorithm.

5 Which Language Should Be Used for the Knowledge Base?

So far we have not said anything about the description logic in which the knowledge base should be written. Algorithm 2 does not make any explicit requirements except that minimal possible consequences should exist. Of course, the whole algorithm only makes sense if it can terminate. In this section we try to find out for which logics this is the case.

The most natural choice for the logic of the knowledge base is $\mathcal{EL}_{\mathrm{gfp}}$. Unfortunately, $\mathcal{EL}_{\mathrm{gfp}}$ is not suitable, because it is not expressive enough to express that an individual is a counterexample to a given GCI $C \sqsubseteq D$. Intuitively, this is because $\mathcal{EL}_{\mathrm{gfp}}$ does not provide any form of negation. For example, it is impossible to state in $\mathcal{EL}_{\mathrm{gfp}}$ that `Henry` from the model from Figure 1 is not an instance of `Mother`. Therefore, it is impossible to state that `Henry` is a counter-example to the GCI `Father` $\sqsubseteq$ `Mother`. Because the expert cannot describe counter-examples the algorithm cannot terminate if $\mathcal{EL}_{\mathrm{gfp}}$ is used for the knowledge base.

5.1 $\mathcal{EL}_{\mathrm{gfp}}$ with Negated Concept Assertions

We have seen that we require at least some weak form of negation in the underlying knowledge base, or else the algorithm cannot terminate. On the other hand, we do not want to make the language of the knowledge base unnecessarily complicated. A simple extension are negated concept assertions.

A *negated concept assertion* is a statement of the form $\neg C(a)$, where C is a concept description. The semantics of negated concept assertions is defined in

a straightforward way. Let $\mathcal{A}$ be an ABox that contains role assertions, concept assertions and negated concept assertions. An interpretation i is a model of $\mathcal{A}$ if and only if for all concept assertions $C(a)$ in $\mathcal{A}$ it holds that $a^i \in C^i$, and for all negated concept assertions $\neg C(a)$ it holds that $a^i \notin C^i$, and for all role assertions $r(a,b)$ it holds that $(a^i, b^i) \in r^i$.

In the setting that we consider in this subsection we are given a background model i. The concept assertions and negated concept assertions occurring in $\mathcal{A}$ shall use $\mathcal{EL}_{\text{gfp}}$-concept descriptions over Σ_0, the signature of i. We do not explicitly use a TBox, but TBoxes are, of course, implicitly present within the $\mathcal{EL}_{\text{gfp}}$-concept descriptions. Allowing (unfoldable) TBoxes explicitly would not result in more expressivity. Obviously in this setting counterexamples do exist and are easy to describe. To turn an individual a into a counterexample to a GCI $C \sqsubseteq D$ we simply need add $C(a)$ and $\neg D(a)$ to the ABox. However, there can still be situations (i. e. background models) where the algorithm cannot terminate.

Consider the background model i depicted in Figure 2. Let the signature be $\mathcal{N}_C = \{P, Q\}$ and $\mathcal{N}_r = \{r\}$. Assume that $\mathcal{A}$ is an ABox that has i as its model. Clearly, if $\mathcal{A}$ is empty then any interpretation is a model and thus any concept description D is a possible consequence of P. In particular this means that $\{x\}^i$ is not minimal among the possible consequences of P.

If $\mathcal{A}$ is not empty, then there is exactly one individual present in $\mathcal{A}$ because of the unique names assumption and because there is only one individual in the background model. We denote this individual by a. Let $\mathcal{A}$ contain the concept assertions $T_1(a), \ldots, T_t(a)$, and the negated concept assertions $\neg F_1(a), \ldots, \neg F_f(a)$, and possibly a single role assertion $r(a,a)$. For every concept description F_k that occurs in a negated concept assertion we can define the Q-depth d_{F_k} of F_k. By d_{F_k} we denote the minimal role depth at which Q appears in F_k. Define $d = 1 + \max_{1 \leq k \leq f} d_{F_k}$.

Now, look at the model ι depicted in Figure 3. The model ι is obtained from i by attaching to x a sequence of nodes v_k, $k \in \{1, \ldots, d\}$ where each node is connected to its successor by the role r. The last of these new nodes v_d is in Q^ι. Clearly, the role assertion $r(a,a)$ holds in ι. All positive concept assertions from $\mathcal{A}$ hold in ι because they hold in i and i is a submodel of ι. All negated concept assertions $N_k(a)$ from $\mathcal{A}$ also hold in ι, because Q occurs in N_k at a role depth less than d, but there is no path of length less than d leading from x to an individual in Q^ι. Therefore ι is a model of $\mathcal{A}$. It holds that $x \in E^\iota$, with $E = (A_E, \mathcal{T}_E)$ where $\mathcal{T}_E$ is defined as

$$\mathcal{T}_E = \{A_E \equiv P \sqcap \exists r.A_E \sqcap \underbrace{\exists r.\exists r \ldots \exists r}_{d \text{ times}}.Q\}.$$

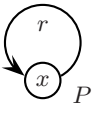


Fig. 2. The model i used in Section 5.1

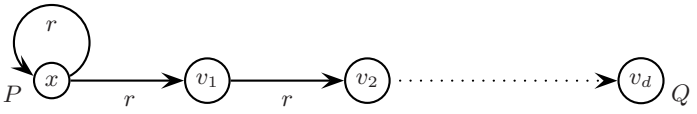


Fig. 3. The model ι used in Section 5.1

Thus E is a possible consequence for P. In particular this proves that $\{x\}^i = P^{ii}$ is not a minimal possible consequence of P.

Now assume that the algorithm has reached a point where $P_k = \{P\}$. The condition required to leave the inner-while loop is that the expert accepts the GCI $P \sqsubseteq D$ where D is a minimal possible consequence for P. We have seen that this can only be the case if $D = P^{ii}$ (Lemma 5). But this can never happen, as for no ABox – be it empty or non-empty – P^{ii} is a minimal possible consequence of P. Hence, for our purposes negated concept assertions are an insufficient extension to $\mathcal{EL}$.

5.2 $\mathcal{EL}$ with $\bot$ and General TBoxes

The bottom concept $\bot$ is a concept constructor whose semantics is defined as $\bot^i = \emptyset$. We suggest to use $\mathcal{EL}$ with the bottom concept $\bot$ and general TBoxes (from now on denoted as $\mathcal{EL}_\bot$) as the DL for the knowledge base. First of all, this logic is a fragment of $\mathcal{EL}^{++}$, a well-supported, tractable DL that is used in many applications such as SNOMED.

$\mathcal{EL}_\bot$ is a minimal extension of $\mathcal{EL}$ in which it is possible to provide counterexamples. Consider for example the model from Figure 1. The model contains a counter-example to the GCI $\texttt{Father} \sqsubseteq \texttt{Mother}$, namely the individual `Henry`. To describe this counter-example in $\mathcal{EL}_\bot$ we have to express that `Henry` is not an instance of `Mother`. We can do this by extending the signature of the knowledge base by adding a new concept name $T_{\texttt{Henry}}$. Then we add the GCI $T_{\texttt{Henry}} \sqcap \texttt{Mother} \sqsubseteq \bot$ to the TBox and $\texttt{Father}(\texttt{Henry})$ and $T_{\texttt{Henry}}(\texttt{Henry})$ to the ABox. This implies that `Henry`, as an ABox-individual, must be an instance of `Father`, yet it cannot be an instance of `Mother`. Notice, that we need to extend the signature of the TBox, in order to describe the counterexample.

Termination is Possible. Regarding termination of Algorithm 2 one can ask two questions. First, is it possible that an expert with an optimal strategy of providing counterexamples can force the algorithm to terminate? And second, is it possible to modify the algorithm such that it terminates, even if the expert uses a suboptimal strategy? While the second question is part of ongoing work, we can give the answer to the first question for $\mathcal{EL}_\bot$.

We have seen in Section 4.2 that Algorithm 2 terminates if and only if it does not get stuck in the inner while loop (Lines 6 to 10). It will leave the inner while loop when $D \equiv (\bigsqcap P_k)^{ii}$ holds. That means, the expert can force the algorithm to terminate if she can come up with an ABox $\mathcal{A}_j$ and a TBox $\mathcal{T}_j$ such that $(\bigsqcap P_k)^{ii}$ is the only minimal possible consequence of $\bigsqcap P_k$.

Let i be the background model that is known to the expert and $\Sigma_0 = (\mathcal{N}_C, \mathcal{N}_r)$ its signature. We prove that enforcing termination is possible by providing a construction for such ABox and TBox from i and $\mathcal{A}_{j-1}$ and $\mathcal{T}_{j-1}$. For every $x \in \Delta_i$ extend the signature Σ_{j-1} by concept names T_x and F_x. Add an individual a_x for every $x \in \Delta_i$. $\mathcal{T}_j$ is obtained from $\mathcal{T}_{j-1}$ by adding statements

$$T_x \sqcap F_x \sqsubseteq \bot \qquad \text{for all } x \in \Delta_i, \tag{3}$$

$$A \sqsubseteq F_x \qquad \text{for all } A \in \mathcal{N}_C \text{ with x } \notin A^i, \tag{4}$$

$$\exists r. \bigsqcap \{F_y \mid y \in xr^i\} \sqsubseteq F_x \qquad \text{for all } r \in \mathcal{N}_r. \tag{5}$$

$\mathcal{A}_j$ is obtained from $\mathcal{A}_{j-1}$ by adding the following statements

$$A(a_x) \qquad \text{for every } x \in \Delta_i \text{ and for every } A \in \mathcal{N}_C \text{ with } x \in A^i, \tag{6}$$

$$r(a_x, a_y) \qquad \text{for every } r \in \mathcal{N}_r \text{ and for all } x, y \in \Delta_i \text{ with } (x, y) \in r^i, \tag{7}$$

$$T_x(a_x) \qquad \text{for every individual } x \in \Delta_i. \tag{8}$$

The concept T_x intuitively represents the properties that x does have while F_x represents the properties that x does not have. In the following we shall prove that for any arbitrary concept description C over the signature Σ_0 the concept C^{ii} is the only minimal possible consequence with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.

Lemma 6. *$\mathcal{A}_j$ and $\mathcal{T}_j$ are a representation of i.*

Proof. We have assumed that $\mathcal{A}_{j-1}$ and $\mathcal{T}_{j-1}$ are representations of the background model i. That means, there is a model ι of $\mathcal{A}_{j-1}$ and $\mathcal{T}_{j-1}$ such that ι restricted to Σ_0 is identical to i. We can further extend ι to a model $\bar{\iota}$ by defining $T_x^{\bar{\iota}} = \{x\}$ and $F_x^{\bar{\iota}} = \Delta_i \setminus \{x\}$. Then $\bar{\iota}$ is a model of $\mathcal{A}_j$ and $\mathcal{T}_j$ (it is simple to check that each of the statements (3) to (8) holds in $\bar{\iota}$). This shows that $\mathcal{A}_j$ and $\mathcal{T}_j$ are a representation of i.

Lemma 7. *Let C be any $\mathcal{EL}_{\text{gfp}}$-concept description over the signature Σ_0. C^{ii} is a possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.*

Proof. We have already seen that $\bar{\iota}$ is a model of $\mathcal{A}_j$ and $\mathcal{T}_j$. Since C and C^{ii} use only the signature Σ_0 (and not the new concept names F_x and T_x, $x \in \Delta_i$) it holds that $C^i = C^{\bar{\iota}}$ and $C^{iii} = (C^{ii})^{\bar{\iota}}$. Lemma 2 states that $C^i = C^{iii}$ and thus $C^{\bar{\iota}} = (C^{ii})^{\bar{\iota}}$. Thus C^{ii} is a possible consequence of C in $\mathcal{A}_i$ and $\mathcal{T}_i$.

Lemma 8. *Let $C = (A_C, \mathcal{T}_C)$ be an $\mathcal{EL}_{\text{gfp}}$-concept description over the signature Σ_0. Let $x \in \Delta_i$ be an individual. If there is a model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ such that $a_x^\iota \in C^\iota$ then $x \in C^i$ holds.*

Proof. Let ι be a model such that $a_x^\iota \in C^\iota$. From Lemma 1 it follows that there is a simulation $\zeta_{C\iota}$ from $\mathcal{T}_C$ to ι such that $(A_C, a_x^\iota) \in \zeta_{C\iota}$. Define ζ_{Ci} as follows:

$$\zeta_{Ci} = \{(B, y) \mid \exists z \in \Delta_\iota : z \notin F_y^\iota \text{ and } (B, z) \in \zeta_{C\iota}\}.$$

To prove that ζ_{Ci} is a simulation from $\mathcal{T}_C$ to i that contains (A_C, x) one must prove (S1), (S2) and $(A_C, x) \in \zeta_{Ci}$. Due to space restrictions we only show the interesting part (S2). Let $r \in \mathcal{N}_r$ be a role name, $(B, y) \in \zeta_{Ci}$ and $E \in Br_{\mathcal{T}_C}$. By definition of ζ_{Ci} there is some $z \in \Delta_\iota$ such that $z \notin F_y^\iota$ and $(B, z) \in \zeta_{C\iota}$. Because $\zeta_{C\iota}$ is a simulation there must be some $\bar{z} \in zr^i$ such that $(E, \bar{z}) \in \zeta_{C\iota}$.

Suppose that for all $\bar{y} \in yr^i$ it holds that $(E, \bar{y}) \notin \zeta_{Ci}$. This implies that $\bar{z} \in F_{\bar{y}}^\iota$ for all $\bar{y} \in yr^i$. But $\mathcal{T}_j$ contains the statement $\exists r.\bigsqcap\{F_{\bar{y}} \mid \bar{y} \in yr^i\} \sqsubseteq F_y$. The above proves $\bar{z} \in (\bigsqcap\{F_{\bar{y}} \mid \bar{y} \in yr^i\})^\iota$ and thus $z \in (\exists r.\bigsqcap\{F_{\bar{y}} \mid \bar{y} \in yr^i\})^\iota \subseteq F_y^\iota$. This contradicts $z \notin F_y^\iota$. We have shown by contradiction that (S2) holds. Together with the omitted steps this proves that ζ_{Ci} is a simulation from $\mathcal{T}_C$ to i such that $(C, x) \in \zeta_{Ci}$. Lemma 1 shows that $x \in C^i$.

Lemma 9. *Let $C = (A_C, \mathcal{T}_C)$ be an $\mathcal{EL}_{\text{gfp}}$-concept description over the signature Σ_0. Let $x \in \Delta_i$. If $x \in C^i$ holds then $\mathcal{A}_j, \mathcal{T}_j \models C(a_x)$*

Proof. We need to show that for any model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ it holds that $a_x^\iota \in C^\iota$. Because of $x \in C^i$ there must be a simulation ζ_{Ci} from $\mathcal{T}_C$ to i containing (A_C, x). Define $\zeta_{C\iota} = \{(B, a_y^\iota) \mid (B, y) \in \zeta_{Ci}\}$. As above, we omit the proofs for (S1) and $(A_C, a_x^\iota) \in \zeta_{C\iota}$. We only prove the interesting step (S2). Let $r \in \mathcal{N}_r$ be a role name, let $(B, a_y^\iota) \in \zeta_{C\iota}$, and let $\bar{B} \in Br_{\mathcal{T}_C}$. $(B, a_y^\iota) \in \zeta_{C\iota}$ implies $(B, y) \in \zeta_{Ci}$. Because ζ_{Ci} is a simulation there is some $\bar{y} \in yr^i$ such that $(\bar{B}, \bar{y}) \in \zeta_{Ci}$. The latter implies that $(\bar{B}, a_{\bar{y}}^\iota) \in \zeta_{C\iota}$. $\bar{y} \in yr^i$ implies that $\mathcal{A}_j$ contains a statement $r(a_y, a_{\bar{y}})$ and therefore $a_{\bar{y}}^\iota \in a_y^\iota r^\iota$. This proves (S2). From Lemma 1 and the fact that $\zeta_{C\iota}$ is a simulation it then follows that $a_x^\iota \in C^\iota$. Since ι was an arbitrary model it follows that $\mathcal{A}_j, \mathcal{T}_j \models C(a_x)$.

Theorem 2. *Let C be any $\mathcal{EL}_{\text{gfp}}$-concept description. C^{ii} is the only minimal possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.*

Proof. Lemma 7 proves that C^{ii} is a possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$. Let D be another possible consequence of C. That means there is a model ι of $\mathcal{A}_j$ and $\mathcal{T}_j$ such that $C^\iota \subseteq D^\iota$. From Lemma 9 it follows that $a_x^\iota \in C^\iota$ for all $x \in C^i$ and thus also $a_x^\iota \in D^\iota$. But then Lemma 8 implies that $x \in D^i$ for all $x \in C^i$, i. e. $C^i \subseteq D^i$. Because most specific concepts are minimal with respect to $\sqsubseteq$ we obtain that $C^{ii} \sqsubseteq D$. This proves that C^{ii} is the the only minimal possible consequence of C with respect to $\mathcal{A}_j$ and $\mathcal{T}_j$.

6 Summary and Open Questions

We have shown that the model exploration algorithm from [6] can be adapted to a new setting. In this new setting, the counter-examples for the rejected GCIs are stored in an ABox instead of a model.

In order to adapt the algorithm we have introduced the notion of minimal possible consequences. We have seen that minimal possible consequences can be used to replace model-based most specific concepts in our new setting. We have presented an exploration algorithm (Algorithm 2) and proved its completeness.

We have pointed out that not all DL languages are suitable to describe counterexamples. It is crucial that this language provides negation or disjointness.

This is ongoing work, and the questions of termination and the existence of minimal possible consequences remain open. Termination has partly been addressed in this paper. We have seen that an expert with an optimal strategy can force the algorithm to terminate if $\mathcal{EL}_\bot$ is used for the knowledge base. However, termination is not guaranteed if the expert uses a different strategy.

Existence of minimal possible consequences, has not been addressed in this work. The author is currently working on a modified tableau algorithm which might be used to compute minimal possible consequences for $\mathcal{EL}_\bot$.

References

1. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proc. of IJCAI 2003, pp. 319–324. Morgan Kaufmann, San Francisco (2003)
2. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. of IJCAI 2003, pp. 319–324. Morgan Kaufmann, San Francisco (2003)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
4. Baader, F., Distel, F.: Exploring finite models in the description logic $\mathcal{EL}_{\mathrm{gfp}}$. LTCS-Report 08-05, Chair for Automata Theory, TU Dresden (2008)
5. Baader, F., Distel, F.: A finite basis for the set of $\mathcal{EL}$-implications holding in a finite model. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 46–61. Springer, Heidelberg (2008)
6. Baader, F., Distel, F.: Exploring finite models in the description logic $\mathcal{EL}_{\mathrm{gfp}}$. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548. Springer, Heidelberg (2009)
7. Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. In: Proc. of the IJCAI 2007. AAAI Press/The MIT Press (2007)
8. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 287–291. Springer, Heidelberg (2006)
9. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, New York (1997)
10. Horrocks, I., Patel-Schneider, P., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web Semantics 1(1), 7–26 (2003)
11. Rudolph, S.: Exploring relational structures via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 196–212. Springer, Heidelberg (2004)
12. Spackman, K.A., Campbell, K.E., Cote, R.A.: SNOMED RT: A reference terminology for health care. J. of the American Medical Informatics Association, 640–644 (1997); Fall Symposium Supplement
13. The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)

On Categorial Grammars as Logical Information Systems

Annie Foret and Sébastien Ferré

IRISA, Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
foret@irisa.fr, ferre@irisa.fr

Abstract. We explore different perspectives on how categorial grammars can be considered as Logical Information Systems (LIS) both theoretically, and practically. Categorial grammars already have close connections with logic. We discuss the advantages of integrating both approaches. We consider more generally different ways of connecting computational linguistic data and LIS as an application of Formal Concept Analysis.

1 Introduction

This paper adresses computational linguistic data, with a focus on categorial grammars, also called *type logical grammars*, that are used for syntax analysis, and are of interest for their lexicalization, their logical formulation, their (logical) semantic interface, and some learnability properties [Kan98]. These grammar formalisms, that refer to ideas by Adjukiewicw in 1935, were refined by Lambek (the grammars can be based on Lambek calculus [Lam58] or on Pregroups [Lam99]). Type logical grammars can be viewed as the assignment of properties to words. The properties are expressed in the underlying logic, depending on the framework version. For example, in the Lambek grammars [Lam58], the logic is a non-commutative (the order of words matter) intuitionistic linear logic, that bears close connections with lambda-calculus, functional application and semantics. As a simplified example, a noun such as "John" can be assigned a basic formula N, and a transitive verb such as "likes" can be assigned a type $N \setminus S / N$, meaning that it lacks a group of type N (a noun) on its left and another group of type N, on its right to produce a sentence (of type S). Parsing in this context amounts to a logic deduction that a type string entails the distinguished type S. With rules similar to Modus Ponens, the concatenation of a transitive verb, such as "likes" with a noun on its right has a type sequence that entails the type of an intransitive verb ($N \setminus S$), because $N \setminus S / N, N$ entails $N \setminus S$; if the string is next concatenated with a noun on its left, we get a sentence, because $N, N \setminus S$ entails S. The construction of a particular grammar for a given language is a difficult task, to cover a large variety of linguistic phenomenon, and yield the appropriate structures via a good property assignment to words.

A type logical grammar is reminiscent of formal contexts with words as objects, and logical types as attributes. The design and control of such grammars

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 225–240, 2010.

can benefit from querying, updating and navigating tools in FCA [FR04]. This has already been practically undertaken for prototypes in a pregroup toolbox (for building and parsing grammars of natural languages) [BFar] where words are the objects of a Logical Information System (LIS) and properties of words are logical formulas. However, logical types were merely represented as strings, which do not account for the logic underlying logical types.

After background sections on Logical Information Systems and on Categorial Grammars, we shall discuss some modelling approaches and different ways of connecting computational linguistic data and LIS. In particular, we shall define a LIS logic that models *pregroup* types, a version of categorial grammars.

2 Logical Information Systems

Logical Information Systems (LIS) are based on Logical Concept Analysis (LCA) [FR04]. LCA is an extension of Formal Concept Analysis that allows to use logical formulas for rich object descriptions and expressive queries.

The LCA framework [FR04] applies to logics with a set-valued semantics similar to description logics [BCM+03]. It is sufficient here to define a logic (see [FR04] for a detailed presentation) as a pre-order of formulas. The pre-ordering is the logical entailment, called *subsumption*: e.g., an interval included in another one, a string matching some regular expression, a graph being a subgraph of another one.

Definition 1 (logic). *A* logic *is a pre-order $\mathcal{L}_T = (L, \sqsubseteq_T)$, where L is a set of formulas, T is a customizable parameter of the logic, and $\sqsubseteq_T$ is a* subsumption *relation that depends on T. The relation $f \sqsubseteq_T g$ reads "f is more specific than g" or "f is subsumed by g", and is also used to denote the partial ordering induced from the pre-order.*

The parameter T helps to take into account domain knowledge that may change over time: e.g., an ontology, a taxonomy. In the following, for simplicity, we consider it as a set of *subsumption axioms* $f \prec g$: e.g., *cat* $\prec$ *animal*, *Quebec* $\prec$ *Canada*. In addition to the logic and its axioms, the *logical context* constitutes the third level of knowledge. It defines a set of objects along with their logical description, and a finite subset of formulas, called *vocabulary*, that is used for navigation.

Definition 2 (logical context). *A* logical context *is a tuple $K = (\mathcal{O}, \mathcal{L}_T, X, d)$, where $\mathcal{O}$ is a finite set of objects, $\mathcal{L}_T$ is a logic, $X \subseteq L$ is a finite subset of formulas called the* navigation vocabulary*, and $d \in (\mathcal{O} \to \mathcal{L}_T)$ is a mapping from objects to logical formulas. For any object o, the formula $d(o)$ denotes the* description *of o.*

Each object is described by a single formula for genericity reasons. Even if a description is often in practice a set of properties, it can also be a sequence of properties or any other data structure. The definition of a vocabulary is necessary

because there is often an infinite set of formulas (e.g., intervals, strings). The choice of a relevant vocabulary depends on both the logic and object descriptions. The elements of the vocabulary are called *features*.

Logical contexts make up the core of Logical Information Systems (LIS) [FR04], so that we need both update and information retrieval operations on them. There are update operations for the addition of an axiom, the creation of a new object along with its description, and the showing of a formula as a navigation feature. The *filling of a context* is the successive addition of a set of objects, defining the updatable part of the context. There are also update operations for removing axioms, modifying the description of objects, removing objects, and hiding formulas.

A key feature of LIS, shared by other concept-based information systems [GMA93, DVE06], is to allow the tight combination of querying and navigation. The principle is that, instead of returning a ranking of all the answers to the query, the system returns a set of query *increments* that suggest to users relevant ways to refine the query, i.e., navigation links between concepts, until a manageable amount of answers is reached. There are two information retrieval operations on logical contexts: one to compute the query answers, and another to compute the query increments from those answers.

A query is a logical formula, and its answers are defined as the extent of this formula, i.e., the set of objects whose description is subsumed by this formula.

Definition 3 (extent). *Let K be a logical context, and $q \in \mathcal{L}$ be a query formula. The* extent *of q in K is defined by $K.ext(q) = \{o \in \mathcal{O} \mid d(o) \sqsubseteq_T q\}$.*

The increments of a query q are the features that are frequent in the extent of q, i.e., the features whose extent shares objects with the extent of q. Those increments are partially ordered by subsumption, and should be presented so to users because this gives a more comprehensive view. For instance, if the vocabulary contains continents, countries, and regions, it is better to display them as a tree rather than a flat list. Moreover, it is not necessary to compute and display all of them at once; continents should be displayed first, and could then be expanded on demand to display countries, etc. These increments provide feedback on the current query and answers, as well as several forms of navigation [Fer09]. LIS have been implemented as a user application, Camelis[1]. It has a dedicated interface showing the current query, the extent and the tree of increments (see a screenshot in Figure 2). It can manage logical contexts with up to 10,000 objects and hundreds of features per object. LIS have been applied to many kinds of data, and most noticeably to the management of a collection of > 5000 photos [Fer09], which can be browsed and updated in terms of time, location, event, persons, and objects. In the following of this paper, screenshots show Camelis where the query box is at the top, the extent is presented as a list of object names at the right, and increments are shown as an expandable tree on the left.

Another important aspect about LIS is genericity w.r.t. the logic. It is not fixed *a priori*, and can be plugged in to cope with application specificities. However,

[1] `http://www.irisa.fr/LIS/ferre/camelis`

designing and implementing a logic is a difficult task that cannot be achieved by application designers in general. This is why we have developed a toolbox of logic components, called *logic functors* [FR04, FR06], that can be assembled into arbitrarily complex logics at a high level. For an application designer, the assembling process is similar to assembling data types in programming (e.g., integers, strings, lists, arrays, maps).

Definition 4 (logic functor). *A* logic functor *is a function $\mathcal{F}$ that takes logics $\mathcal{L}_1, ..., \mathcal{L}_n$ as arguments ($n \geq 0$), returns a composed logic $\mathcal{L}_T = \mathcal{F}(\mathcal{L}_1, ..., \mathcal{L}_n)$. As for any logic, one has $\mathcal{L}_T = (L, \sqsubseteq_T)$, but L (resp. $\sqsubseteq_T$) is function of the sets of formulas (resp. subsumption) of arguments logics.*

Thanks to annotations attached to logic functors by their designers, composed logics can be automatically checked for correctness w.r.t. semantics. Another advantage of logic functors is to address the well-known trade-off between efficiency and genericity of logical reasoning. The efficiency comes from the specificity of functors that allows to use dedicated data structures and algorithms, and the genericity comes from the ability to compose them. This is similar to natural languages where a finite number of different words can be combined in an infinity of sentences. Of course, the expressivity is limited by the available functors (resp. words), but new functors (resp. words) can always be invented.

LogFun[2] is a toolbox of logic functors [FR06] that can be used in Camelis. It contains a few dozens of logic functors, some of which are grouped in the following categories:

1. *Concrete domains*: **Int** (integers), **Float** (floating-point numbers), **Time** (hours, minutes, seconds), **Date** (day, month, year), **Char** (characters, and classes such as letters, vowels, digits), **String** (strings, and string patterns such "contains", "begins with"), **Atom** (the classical atoms), **Taxo** (custom taxonomies);
2. *Algebraic structures*: **Unit** (null value), **Top**(L_1) (add a top formula), **Prod**(L_1,L_2) (pairs), **Sum**($\{L_k\}_k$) (alternative), **Interval**(L_1) (intervals), **Enum**(L_1) (enumerations), **Segment**(L_1) (segments, e.g., a period of time), **Multiset**(L_1) (conjunctive multisets), **Motif**(L_1) (complex motifs over sequences, as used in bioinformatics), **Option**(L_1) = Sum(Unit,L_1) (optional value), **List**(L_1) = Top(Option(Prod(L_1,List(L_1)))) (lists, and list prefixes), **Vector**(L_1) = Option(Prod(L_1,Vector(L_1))) (vectors), **Tree**(L_1) = Top(Prod(L_1,List(Tree(L_1)))) (n-ary trees, and tree prefixes), **Prop**(L_1) (Boolean propositions);
3. *Dedicated logics*: complex combinations of logic functors represent the types of various programming languages: **Java**, **Caml**, **Mercury**.

Note how some logic functors are defined from other functors. Sometimes, those definitions are recursive like in **List**, **Vector**, and **Tree**, thus allowing the representation of recursive data structures. In Section 5, we define a dedicated logic to represent *pregroup types*, in order to describe words, phrases, and sentences.

[2] `http://www.irisa.fr/LIS/ferre/logfun/`.

3 Categorial Grammars

We first consider categorial grammars in general: these are lexicalized grammars, involving an alphabet Σ, a set of *types* Tp that is usually generated from a set of *primitive types* Pr and given connectives, and a system of rules on types (a "type calculus") defining a derivation relation $\vdash$ on types.

3.1 Categorial Grammars and Their Languages

Definition 5 (Categorial grammar). *Given a set Tp called the set of types :* a categorial grammar *is a structure $G = (\Sigma, I, S)$ where: Σ is a finite alphabet (the words in the sentences); $I : \Sigma \mapsto \mathcal{P}^f(Tp)$ is a function that maps a finite set of types from each element of Σ (the possible categories of each word); $S \in Pr$ is the* main type *associated to correct sentences.*

Given a relation on Tp^* called the derivation relation on types : a sentence $v_1 \dots v_n$ then belongs to the language of G, *written $\mathcal{L}(G)$, provided its words v_i can be assigned types X_i whose sequence $X_1 \dots X_n$ derives S according to the derivation relation on types.*

3.2 AB and Lambek Grammars

We now give the deduction rules for classical categorial grammars, also known as *AB-grammars*.

Definition 6. *An* AB-grammar *is a categorial grammar $G = (\Sigma, I, S)$, such that its set of types is constructed from Pr, using two binary connectives $/$, $\setminus$:*

$Tp ::= Pr \mid Tp \setminus Tp \mid Tp \,/\, Tp$

and its language is defined using two deduction rules:

$A, A \setminus B \vdash B$ *(Backward elimination, written $\setminus_e$)*

$B \,/\, A, A \vdash B$ *(Forward elimination, written $/_e$)*

For example, using $\setminus_e$, the string of types $(N, N \setminus S)$ associated to "John swims" entails S, the type of sentences. Another typical example is $(N, ((N \setminus S) \,/\, N), N))$ associated to "John likes Mary", where the right part associated to "likes Mary" $(((N \setminus S) \,/\, N), N)$ entails $(N \setminus S)$, which combined on the left with the type of "John" : $(N, N \setminus S)$ entails S, are above. AB-grammars are the basis of a hierarchy of type-logical or Lambek-like grammars. These extensions usually provide more type flexability, and turn the type calculus into an actual (substructural) logic system, in the range of linear and intuitionnistic logics. The associative Lambek calculus has been introduced in [Lam58], we refer to [Bus97] for details on this calculus and its non-associative variant.

3.3 Pregroups

The pregroup formalism has been introduced recently [Lam99] as a simplification of Lambek calculus [Lam58]. It is considered theoretically for the syntax modeling of various natural languages and also practically with parsers [DP05, Oeh04, Béc07] and sofware tools for grammar construction [BF09].

Definition 7. *A* pregroup *(or* PG *in short) is a structure* $(P, \leq, \cdot, l, r, 1)$ *such that* $(P, \leq, \cdot, 1)$ *is a partially ordered monoid*[3] *and* l, r *are two unary operations on* P *that satisfy for all* $a \in P$: $a^l a \leq 1 \leq a a^l$ *and* $a a^r \leq 1 \leq a^r a$.

Example 1. Groups are a special case of pregroups, for which the left and right adjoint are the same (the group inverse).

Algebraic properties and iterated adjoints. In a pregroup, we have :
$(XY)^l = Y^l X^l$ and $(XY)^r = Y^r X^r$;
$(X^l)^r = X$ and $(X^r)^l = X$
$1^l = 1^r = 1$
Iterated left and right adjoints can be written using integer exponents by:
$p^{(n-1)} = (p^{(n)})^l$ and $p^{(n+1)} = (p^{(n)})^r$, we write p for $p^{(0)}$.
Note also that if $X \leq Y$ then $Y^l \leq X^l$ and $Y^r \leq X^r$ (order-reversing)

Definition 8 (Pregroup grammars and languages). *A pregroup grammar (PG-grammar in short) based on a finite poset* $(P, \leq)$ *is a categorial grammar* $G = (\Sigma, I, s)$*, taking as set of types:*

$$Tp{=}Cat_{(P,\leq)}{=}\{p_1^{(i_1)} \cdots p_n^{(i_n)} \mid 0 \leq k \leq n, p_k \in P \wedge i_k \in \mathbb{Z}\}$$

The language $\mathcal{L}(G)$ *of a PG-grammar* $G = (\Sigma, I, s)$*, based on a finite poset* $(P, \leq)$ *is the set of strings, that can be assigned a string of types deriving* s *in the free pregroup on* $(P, \leq)$*:*
$\mathcal{L}(G) = \{v_1 \dots v_n \in \Sigma^* : \forall i, 1 \leq i \leq n, \exists X_i \in I(v_i)$ *such that* $X_1 \dots X_n \leq s\}$

Buszkowski has proposed a sequent calculus, written $\mathcal{S}^{Adj}$ below, that characterizes the free pregroup derivation relation, in which the cut rule can be eliminated [Bus01a]

Definition 9 (The Sequent calculus of Adjoints $\mathcal{S}^{Adj}$). *Let* $(P, \leq)$ *be an ordered set of primitive types, system* $\mathcal{S}^{Adj}$ *is made of the following rules, where* $p, q \in P$, $n, k \in \mathbb{Z}$ *and* $X, Y, Z \in Cat_{(P,\leq)}$*:*

$$X \leq X \ \ (Id) \qquad \frac{X \leq Y \quad Y \leq Z}{X \leq Z}\,(Cut)$$

$$\frac{XY \leq Z}{X p^{(n)} p^{(n+1)} Y \leq Z}\,(A_L) \qquad \frac{X \leq YZ}{X \leq Y p^{(n+1)} p^{(n)} Z}\,(A_R)$$

$$\frac{X p^{(k)} Y \leq Z}{X q^{(k)} Y \leq Z}\,(IND_L) \qquad \frac{X \leq Y q^{(k)} Z}{X \leq Y p^{(k)} Z}\,(IND_R)$$

$q \leq p$ *if* k *is even, and* $p \leq q$ *if* k *is odd*

[3] We briefly recall that a *monoid* is a structure $< M, \cdot, 1 >$, such that $\cdot$ is associative and has a neutral element 1 ($\forall x \in M : 1 \cdot x = x \cdot 1 = x$). A partially ordered monoid is a monoid $M, \cdot, 1)$ with a partial order $\leq$ that satisfies $\forall a, b, c$: $a \leq b \Rightarrow c \cdot a \leq c \cdot b \wedge a \cdot c \leq b \cdot c$.

The languages generated by PG-grammars based on finite posets are the context-free languages [Bus01b], (without the empty string).

We carry on with a previous formal example.

Example 2. Let $\Sigma = \{a, b\}$ and $G'_{PG} = (\Sigma, I', s)$, with assignments of types (on $Pr = \{s, x\}$):

$$G'_{PG} \quad \boxed{\begin{array}{l} a \mapsto sx^l \\ \quad \mapsto sx^l s^l \\ b \mapsto x \end{array}}$$

We have $\mathcal{L}(G'_{PG}) = \{a^n b^n \mid n \geq 1\}$

3.4 Interpreting One Type System in Another

Some links between categorial frameworks can be reflected in the logical system. We explain such a link, that is also illustrated in next example.

Free pregroup interpretation. We associate with each formula C of the Lambek Calculus L or its non-associative version NL, an element $[\![C]\!]$ in a free pregroup FP defined by:

$[\![A]\!] = A$ if A is a primitive type,
$[\![C_1 \setminus C_2]\!] = [\![C_1]\!]^r [\![C_2]\!]$
$[\![C_1 \,/\, C_2]\!] = [\![C_1]\!][\![C_2]\!]^l$
$[\![C_1 \bullet C_2]\!] = [\![C_1]\!][\![C_2]\!]$.

The notation extends to sequents by:

$[\![A_1, \ldots, A_n]\!] = [\![A_1]\!] \cdots [\![A_n]\!]$.

The following property states that the free pregroup FP is a model for L (hence for NL):

if $\Gamma \vdash_L C$ then $[\![\Gamma]\!] \leq_{FP} [\![C]\!]$.

Note however that the converse does not hold [Bus01b]:

$(a.b) \,/\, c \not\vdash a.(b \,/\, c)$ with $[\![(a.b) \,/\, c]\!] = [\![a.(b \,/\, c)]\!] = a.b.c^l$

$(p \,/\, ((p \,/\, p) \,/\, p)) \,/\, p \not\vdash p$ with $[\![(p \,/\, ((p \,/\, p) \,/\, p)) \,/\, p]\!] = pp^{ll}p^{ll}p^l p^l \leq p$

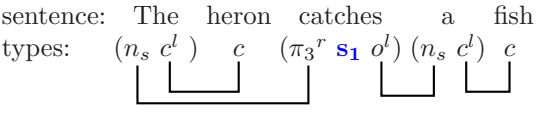
It is of interest to keep in parallel a type and its interpretation or projection in another framework, for example to guide and control the design of a grammar or to compare formalisms.

3.5 A Detailed Example

We now detail a linguistic example as a pregroup grammar, and a possible link with other type grammars.

Example 3. Let $P_1 = \{c, n_s, \pi_3, s_1, o, s, \overline{s}\}$ denote a set primitive types ordered by $n_s \leq \pi_3, n_s \leq o, s_1 \leq s, s \leq \overline{s}$, let $Tp = Cat_{(P_1, \leq)}$ denote the set of types. We consider the following PG-grammar $G_1 = (\Sigma_1, I_1, s)$ based on $(P_1, \leq)$ for a fragment of English, such that: $\Sigma_1 = \{a\ ,\ The\ ,\ catches\ ,\ heron\ ,\ fish\}$ $I_1(The) = I_1(a) = \{n_s c^l\}$, $I_1(heron) = I_1(fish) = \{c\}$, $I_1(catches) = \{\pi_3{}^r s_1\ o^l\}$

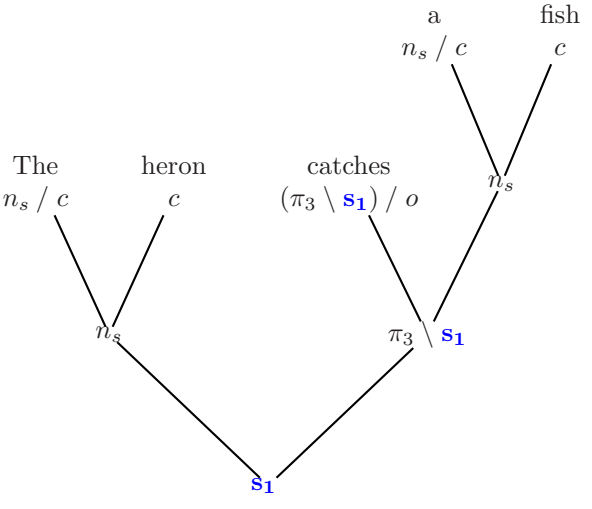
(the type assignment is inspired from [BL01, Lam06]). The following sentence belongs to $\mathcal{L}(G_1)$, because the string of types derives $s_1 \leq s$ (the type assignments are below each word, underlinks indicate contractions of types, such as $X\, c^l\, c\, Y \leq X\, Y$ or $X\, o^l\, n_s\, Y \leq X\, Y$, because of the preorder relation $n_s \leq o$):



using primitive types and order postulates as follows:

	c = count noun	(heron, fish)
$n_s \leq \pi_3$	n_s = singular noun phrase	(John)
	π_k = k^{th} personal subject pronoun	(π_3=he/she/it)
$n_s \leq o$	o = direct object	
$s_1 \leq s \leq \overline{s}$	s_1 = statement in present tense	
	s = declarative sentence	(the tense does not matter)
	$\overline{s}$ = indirect sentence	

Using the $[\![.]\!]$ translation to Lambek calculus L, this sentence is also parsed in L:



where types n_s, π_3, o ... are to be replaced with complex types such that: $n_s \vdash \pi_3$, $n_s \vdash o$, and $s_1 \vdash s \vdash \overline{s}$.

4 Modelling Approaches

Before we detail different approaches, we give a general view on integrating perspectives. We shall discuss some possible perspectives relating LIS and some aspects of NLP (Natural Language Processing), with a focus on categorial grammars.

The following table sums up what can be taken as object, and what as property. A syntax-pos tag is a general syntactic category of words, such as Determiner, Noun, Verb. A macrotype is a more refined syntactic category, such as "transitive verb in present tense", "plural noun".

Object	Property	Relevance (-/+)
word	syntax-pos tag	- weak no actual preorder/hierarchy
word	categorial type	+ logic preorder/hierarchy
type	word	- similarity on words, lemmatization ... ; semantic subsumption...
macrotype	categorial type	+ logic preorder/hierarchy
sequence of words	categorial type	+ logic preorder/hierarchy

In fact, LIS do not force to separate these views, it is quite flexible w.r.t. heterogeneity and partial knowledge; for example, we can consider words and sequences of words together, even if the sequences are not all typed in the sense of parsing (whereas words are typed). This is illustrated in figure 1, where various informations are gathered for words, sentences and types (indicating in particular an example of use of this assignment to a word in a sentence). Such a combination may be fruitful to explore words in context (corpus fragments).

Such modellings of objects and properties (1) can be discussed with respect to some other aspects: formal context (2), set of features (3), navigation links (4), clusters (5), association rules (6).

4.1 Words as Objects and POS Tags as Properties

A direct application is to consider that the information attached to words, such as part-of-speech (POS) tags, are like attributes. For example, when common POS tags are associated to words, as in figure 2, a typical use is as in figure 3: the query selects the objects, whose property *mot* (word) contains "grand" (french for "big"), and whose property *cat* contains "adv" (for adverbs, with different subclasses in this context).

4.2 Macrotypes as Objects

This has been used as a stage to control a conversion of a lexicon in one format, into a lexicon in another format, using macrotypes [BF09].

Below is a typical line of a file describing a context, for a conversion of tags in Lefff [4] (a large-scale morphological and syntactic lexicon for French), here a macrotype "det" is intended to be converted into a categorial type "d"

[4] Lefff stands for: "Lexique des Formes Fléchies du Français / Lexicon of French inflected forms"[see http://alpage.inria.fr/~sagot/lefff-en.html]

Fig. 1. A toy grammar, with additional informations, as a LIS context

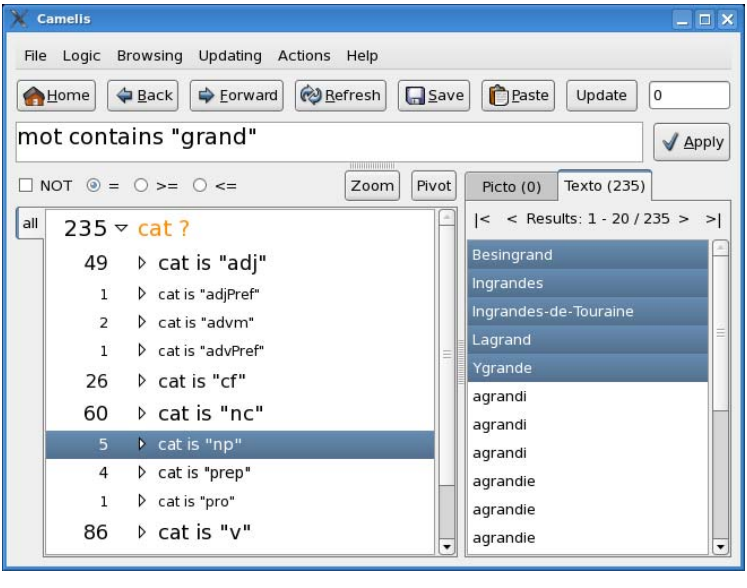


Fig. 2. Words with their categories, from a fragment of Lefff3

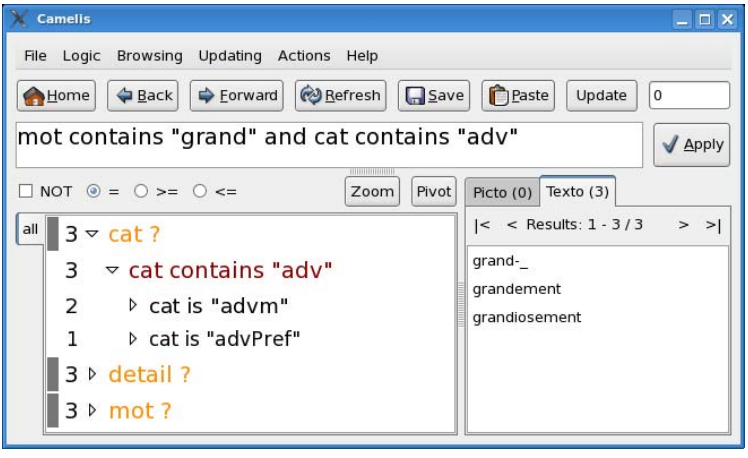


Fig. 3. Words with their categories, from a fragment of Lefff3: selection of adverbs containing "grand"

(`type is "d."`) while keeping another information: the Lambek version of the macrotype (`chaineL is "n/n"`).

```
mk "det" class is "macro", type is "d.", chaineL is "n/n;"
```

4.3 Categorial Types and Concepts

We now give a toy example, to illustrate an application to the construction and maintenance of categorial grammars.

Example 4. We consider two mini-grammars G_{fr} G'_{fr}, that differ in the treatment of type t_0.

$$G_{fr} = \begin{cases} w_0, w'_0 \mapsto t_0 \\ w_1 \mapsto t_1, t_2 \\ w_2 \mapsto t_0 \end{cases} \text{ where } \begin{cases} t_0 = N \\ t_1 = N^{(1)}S \\ t_2 = N^{(1)}SN^{(-1)} \end{cases} \text{ s.t. } \begin{cases} [\![t_0]\!] = N \\ [\![t_1]\!] = N \setminus S \\ [\![t_2]\!] = N \setminus S \,/\, N \end{cases}$$

we take $w_0 = John$, $w'_0 = he$ $w_1 = eats$, $w_2 = apples$, in which case, the sequence "$w_0.w_1.w_2$" is a sentence that receives type S after a logical derivation. This grammar is described by: $d(w_0) = d(w'_0) = t_0; d(w_1) = t_1 \wedge t_2; d(w_2) = t_0$

Let G'_{fr} be defined as G_{fr} except for w'_0 such that $w'_0 \mapsto t'_0$ with axiom $t_0 \leq t'_0$ (note that $w_0 \mapsto t_0 \leq t'_0$).

The advantage of G'_{fr} over G_{fr} is to reject $w_2.w_1.w'_0$; this is because $t_0 \leq t'_0$, but not the converse, which prevents the type cancellation on $w_1.w'_0$ ("he" cannot be the object of a verb, whereas "apples" can).

These words appear with a few others in figure 4.

Another view is to take more advantage of the logic nature of categorial grammars. This point is developed in next section.

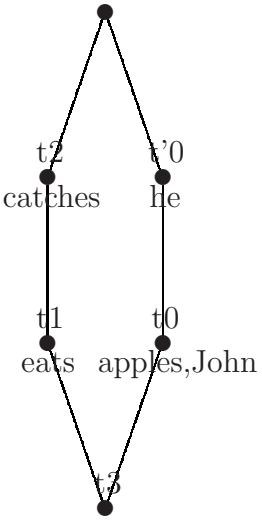


Fig. 4. Concept Lattice of a toy categorial grammar

5 Categorial Grammars as Logical Contexts

Pregroup types are a good example of representations that cannot easily be split into binary attributes. From Definition 3.3, a pregroup type has the form $p_1^{(i_1)} \dots p_n^{(i_n)}$. In other words, it is a string of simple types, where each simple type is made of a basic type and an exponent representing iterated adjoints.

In order to apply Camelis to the organisation, search and exploration of a collection of words, group of words and sentences according to their pregroup types, we need to define a logic for representing and reasoning on them. First, we have to take into account the fact that basic types form a partially ordered set. If we search for words having type o (object), we should retrieve words having type n (nouns) because of the axiom $n \leq o$ (nouns can be used as objects). To this purpose, we choose the logic functor **Taxo**, which is dedicated to the representation of customizable taxonomies, i.e., partially ordered sets of terms. Here, terms are the basic types, and the partial ordering is specified by users through axioms. Second, we have to take into account the ordering and adjacency of simple types. If we search for words whose type starts with $\pi^{(1)}s$ (verbs), we should retrieve words with types such as $\pi_3^{(1)}s_1$ (intransitive verbs conjugated for the third person, and at present tense), or $\pi_1^{(1)}s_1\ o^{(-1)}$ (transitive verbs conjugated for the first person, at present tense). The logic functor **Motif** has been designed for handling such patterns over strings and sequences, and was originally applied to biological sequences [FK04]. This logic functor is parameterized by a logic for the sequence elements, here simple types. As a simple type is a pair made of a basic type and an integer exponant, the logic

for simple types can be defined as Prod(Taxo,Int). Finally, a logic for pregroup types can be defined by simply composing logic functors out of our toolbox, as follows.

Pregroup = Motif(Prod(Taxo,Int))

The predefined syntax parser and printer are overloaded to the standard notation of pregroup types. Pregroup types used for describing words are distinguished from patterns by enclosing them in square brackets. The type $[s]$ represents a sentence, whereas the type pattern s represents any word that generates a sentence given appropriate arguments (e.g., a verb). A prefix (resp. suffix) pattern can be expressed by using only the opening (resp. closing) bracket. The type pattern $[\pi^{(1)}s$ represents verbs, both transitive and intransitive, for any person, and at any tense.

In the application, the logic **Pregroup** is used as the value domain of the distinguished attribute `pgtype` (see screenshots).

In figure 5, we get words that can be seen as "objects", however this requires the use of axioms, because no word has type "o" (indeed, the query `type is "o"` return no object) whereas "o" is "required" by transitive verbs on their right hand side to yield a sentence. The highlighting shows that four of these words have in fact "n" as an explicit type. In figure 6, two transitive verbs are selected because they match the pgtype in the query whereas it is not in their explicit assignment. The pgtype of the two words specifies the tense and the

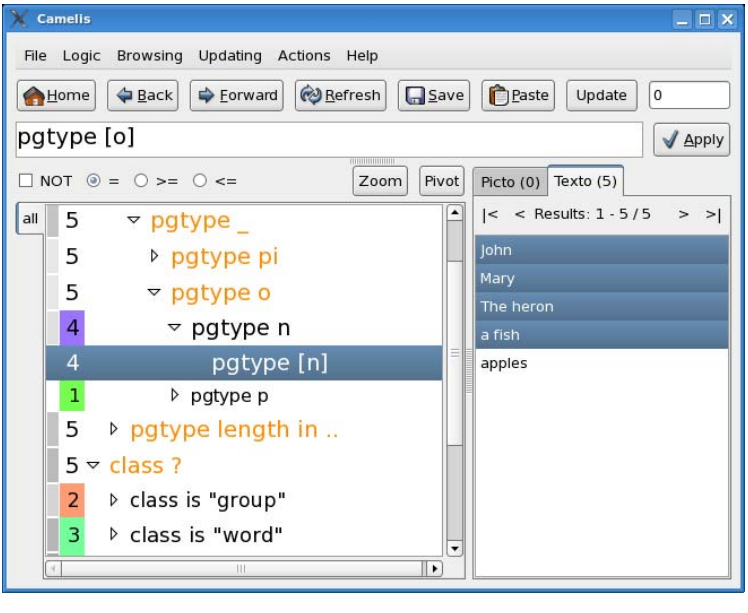


Fig. 5. A toy grammar as a LIS context: `pgtype [o]` vs `pgtype [n]`

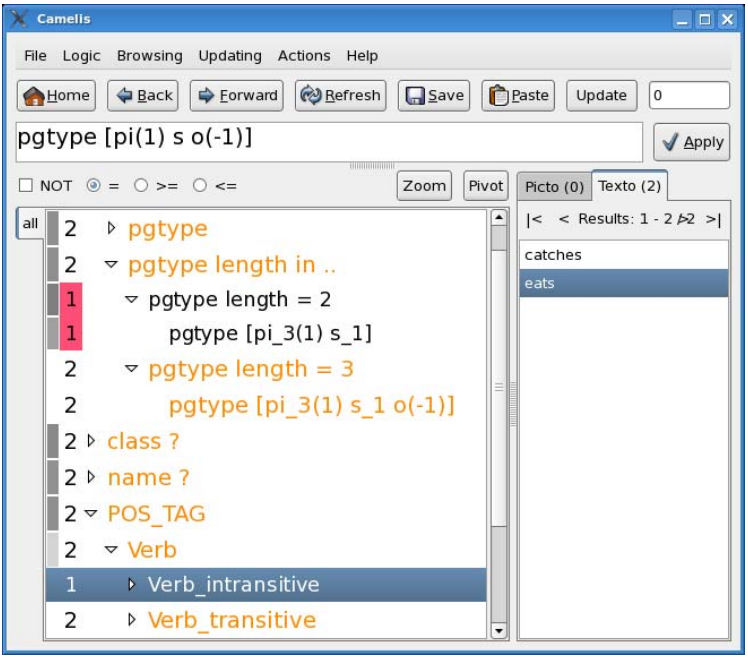


Fig. 6. A toy grammar as a LIS context: selection of transitive verbs

person, while the query does not. The highlighted verb is also intransitive, as exhibited by the increments.

The situation is similar for the query: $pgtype[s]$, in a LIS context with sentences as objects, following a categorial pregroup assignment as well. Using the same toy grammar as previously, this query returns two sentences "John eats apples" and "The heron catches a fish", with property $pgtype[s_1]$; this is because the axiom reflecting the pregoup logic ($s_1 \leq s$) has been used to answer this query.

A further use of categorial logic can be made, implementing completely the pregroup calculus $\leq_P G$ as the subsumption relation between types assigned to words (this can also hold for groups of words, or sentences, using this $\leq_P G$ derivation). Another step is to consider the pregroup calculus as a logic functor, so as to combine it with other logic-calculus (in the sense of LIS). This has been studied in [For07]. The use of general axioms in related formalisms is discussed by [Bus02]. In particular, in the associative Lambek calculus, this may lead to an undecidable extension, whereas this addition of axioms preserves polynomial decidability in the non-associative version.

6 Conclusion

This paper has exposed how categorial grammars (such as pregroups) can be considered as Logical Information Systems (LIS) both theoretically, and

practically. On the one hand, the lexicalized nature of these grammars enables to consider them as a kind of dictionary attaching properties to words; this gives rise to a first modelling via attributes. On the other hand, the logical nature of this linguistic framework enables to go beyond this first modelling, and take full advantage of Logical Concept Analysis, with better representation and propagation of properties.

On the computational linguistic side, such a combination is useful for the creation, control and query of linguistic grammars and prototypes such as the construction of pregroup grammars [BF09]. On the one hand, prototypes have been created manually in a controlled way, with mixed information as LIS contexts. On the other hand automatic conversions between LIS and grammars–possibly large as those obtained from Lefff for French–are helpful. On the logical information system side, the case of categorial grammars illustrates the strength and elegance of composing and defining logic functors .

This study suggests some extensions of LIS, that would be useful for the design of categorial grammars, and linguistic data in general. For example, considering macro-types as intermediary object/properties, we propose to build a three-level LIS, or more generally *cascading LIS* as a finite sequence of logical information systems, where the set of objects of a context is part of the logic of the previous context.

Other future perspectives concern the extension of (1) the kind of objects and informations (heterogeneous data, XML structured sentences), and (2) the query language and logic functors (in particular a pregroup functor based on pregroup inference), as well as (3) ways of integrating learning approaches for augmenting algorithmically the linguistic information.

References

[BCM+03] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)

[Béc07] Béchet, D.: Parsing pregroup grammars and Lambek calculus using partial composition. Studia Logica 87(2/3) (2007)

[BF09] Bechet, D., Foret, A.: Ppq: a pregroup parser using majority composition. In: Proceedings of Parsing with Categorial Grammars, ESSLLI workshop, Bordeaux, France, July 20-24 (2009)

[BFar] Béchet, D., Foret, A.: A pregroup toolbox for parsing and building grammars of natural languages. Linguistic Analysis Journal 36 (to appear)

[BL01] Bargelli, D., Lambek, J.: An algebraic approach to french sentence structure. In: de Groote, P., Morrill, G., Retoré, C. (eds.) LACL 2001. LNCS (LNAI), vol. 2099, p. 62. Springer, Heidelberg (2001)

[Bus97] Buszkowski, W.: Mathematical linguistics and proof theory. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language, ch. 12, pp. 683–736. North-Holland Elsevier, Amsterdam (1997)

[Bus01a] Buszkowski, W.: Cut elimination for the lambek calculus of adjoints. In: New Perspectives in Logic and Formal Linguisitics, Proceedings Vth ROMA Workshop, Bulzoni Editore (2001)

[Bus01b] Buszkowski, W.: Lambek grammars based on pregroups. In: de Groote, P., Morrill, G., Retoré, C. (eds.) LACL 2001. LNCS (LNAI), vol. 2099, p. 95. Springer, Heidelberg (2001)

[Bus02] Buszkowski, W.: Lambek calculus with nonlogical axioms. In: Language and Grammar. Studies in Mathematical Linguistics and Natural Language, pp. 77–93. CSLI Publications (2002)

[DP05] Degeilh, S., Preller, A.: Efficiency of pregroup and the french noun phrase. Journal of Language, Logic and Information 14(4), 423–444 (2005)

[DVE06] Ducrou, J., Vormbrock, B., Eklund, P.W.: FCA-based browsing and searching of a collection of images. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 203–214. Springer, Heidelberg (2006)

[Fer09] Ferré, S.: Camelis: a logical information system to organize and browse a collection of documents. Int. J. General Systems 38(4) (2009)

[FK04] Ferré, S., King, R.D.: BLID: an application of logical information systems to bioinformatics. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 47–54. Springer, Heidelberg (2004)

[For07] Foret, A.: Pregroup calculus as a logical functor. In: Leivant, D., de Queiroz, R. (eds.) WoLLIC 2007. LNCS, vol. 4576, pp. 147–161. Springer, Heidelberg (2007)

[FR04] Ferré, S., Ridoux, O.: An introduction to logical information systems. Information Processing & Management 40(3), 383–419 (2004)

[FR06] Ferré, S., Ridoux, O.: Logic functors: A toolbox of components for building customized and embeddable logics. Research Report RR-5871, INRIA, 103 p. (March 2006)

[GMA93] Godin, R., Missaoui, R., April, A.: Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. International Journal of Man-Machine Studies 38(5), 747–767 (1993)

[Kan98] Kanazawa, M.: Learnable Classes of Categorial Grammars. Studies in Logic, Language and Information. Center for the Study of Language and Information (CSLI) and The European association for Logic, Language and Information (FOLLI), Stanford, California (1998)

[Lam58] Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly 65 (1958)

[Lam99] Lambek, J.: Type grammar revisited. In: Lecomte, A., Perrier, G., Lamarche, F. (eds.) LACL 1997. LNCS (LNAI), vol. 1582, p. 1. Springer, Heidelberg (1999)

[Lam06] Lambek, J.: Pregroups and natural language processing. The Mathematical Intelligencer 28(2) (2006)

[Oeh04] Oehrle, R.: A parsing algorithm for pregroup grammars. In: Moortgat, M., Prince, V. (eds.) Proc. of Intern. Conf. on Categorial Grammars, Montpellier (2004)

Describing Role Models in Terms of Formal Concept Analysis

Henri Mühle[1] and Christian Wende[2]

Technische Universität Dresden
[1]Institut für Algebra
Henri.Muehle@tu-dresden.de
[2]Lehrstuhl für Softwaretechnologie
c.wende@tu.dresden.de

Abstract. In the past years Software Engineering has experienced several difficulties in modularising crosscutting aspects, like shared, dynamic or scattered behavior of object-oriented systems. One approach to overcome these difficulties is to encapsulate such behavior in separate modules, called role models. Role composition provides means to compose coherent, executable software systems from such role models.

This paper focuses on creating a concept-based framework for representing role models. Applying several order-theoretic theorems, Formal Concept Analysis allows for checking the role models and role model composition for consistency and analysing quantitative characteristics of the system design, like size estimation. Another benefit is the ability of Formal Concept Analysis to visualize data and their relations. This provides mechanisms for tracing the lifecycle of role-playing objects at run-time and, thus, for learning about role changes and relations between roles.

Keywords: Formal Concept Analysis, Role Modeling, Concept-driven Framework, System Representation.

1 Introduction

During the past twenty years, Software Engineering has evolved from procedural development towards more and more conceptual approaches ranging from modular design to object-oriented development. However, it has turned out that certain aspects cannot be handled properly. E. g. if the same behaviour is scattered throughout the code or has to be executed right before or right after a certain method is called, standard object-orientation needs to work this around using code duplication over and over again. Another issue that is to be addressed is the dynamic sharing of certain behavior among different objects. Whenever there exists a kind of behavior that can be accessed by different classes, it appears useful to encapsulate this behavior in a particular object. Due to inheritance restrictions it is not always possible to solve this problem in standard object orientation. Since such a kind of behavior can be interpreted as *role behavior* it appeared necessary to extend object orientation towards role orientation [7].

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 241–254, 2010.

Role modeling, however, needs to cope with several difficulties, such as checking for consistency of the role models or representing the system's run-time state. Our approach provides a theoretical formalization of role models and role model composition and, thus, contributes to a better understanding of what is necessary to solve these problems. This helps the development of more complete and robust role modeling approaches than are available at the moment.

In this paper, we thus contribute (1) a static representation for role models and role model composition using Formal Concept Analysis, (2) a representation of the dynamic run-time state of a role-based software system using *template contexts* and (3) approaches for the exploitation of these formal representations to leverage role modeling.

In Sect. 2 we give a short introduction to both – role modeling and Formal Concept Analysis. Then, we show in Sect. 3 how to statically represent basic role models and role composition using appropriate formal contexts. We extend this approach for representing the run-time state of role-based software systems. Afterwards, in Sect. 4 we discuss the benefits of using FCA for role modeling and in Sect. 5 we give an outlook towards future work and conclude our approach.

2 Delivering the Basics

As introduced in [7] a "**role model** is an object-oriented model of an object structure and represents a bounded part of the real world or of an interesting concept. It models patterns of collaborating objects as a similar structure of collaboration roles" [7, p. 71]. In other words, "a role model is the description of a (possibly infinite) set of object collaborations using role types" [8, p. 3].

A definition for roles themselves can amongst others be found in [5]: "A role of an object is a set of properties which are important for an object to be able to behave in a certain way expected by a set of other objects." In collecting various definitions of the concept *role*, however, Friedrich Steimann concludes that there are three possible views onto this concept: "roles as named places of a relationship, roles as a form of generalization and/or specialization, and roles as separate instances joined to an object" [10, p. 5].

In our approach we will more or less refer to all three views. Since roles always appear in a context, or in Reenskaug's words a collaboration [7, p. 64] – which is even more comprehensible, due to the terminology from Formal Concept Analysis – they need to be reckoned as named places in a relationship. Furthermore, roles shall always extend or at least modify the behavior of an object, thus they kind of specialize this object. And last but not least, a role shall be regarded as a separate instance for making their application more flexible.

To the best of our knowledge, Steimann was the first author who introduced an approach to role modeling using partial orders. He contributed a modeling language, called Lodwick, which basically maintains a type hierarchy $(N, \leq_{NN})$, consisting of a set of **natural types** N along with an inheritance hierarchy $\leq_{NN}$, a role hierarchy $(R, \leq_{RR})$, consisting of a set of **role types** R along with an inheritance hierarchy $\leq_{RR}$ as well and a relation $<_{NR} \subseteq N \times R$, describing which

natural type is able to play which role type, which he calls role-filler relationship. Another important fact is the inheritance of the role-filler relationship. Steimann states, "that a type n fills a role r if there is a supertype n' of n that fills a subrole r' of r" [10, p. 13]. This means that each supertype of a given natural type n plays at least the same roles as n does. Likewise, n is able to play with every role type r each subrole of r. LODWICK furthermore allows dynamic modeling of instances and is able to represent the whole lifecycle of a role-modeled system. For a detailed introduction, see [10, pp. 11-16].

The role modelling used in our approach is inspired by the role-oriented programming language ObjectTeams/Java [4]. ObjectTeams introduces a package-like data structure, called team, to denote role models. Teams are designed as containers for role classes which can be instantiated themselves. Hence, they fulfill the same functionality as Reenskaug's collaborations. Furthermore, roles themselves are distinguished in their "playability". Roles that can be played directly by natural types (or as in ObjectTeams/Java: **base types**) are called **bound roles**, while roles which cannot, are called **unbound roles**. However, in this paper we will focus only on bound roles since the benefits of role modeling stem from the composition towards "ordinary" type hierarchies. Unbound roles do not occur in this composition, but may be useful for abstraction or interaction between other roles and can in our approach be regarded as attributes of bound roles. The binding of roles in ObjectTeams/Java is specified via a **role-play relation** which corresponds to Steimann's role-filler relationship.

Example 1. One of the greatest contributions of role modeling is the ability to encapsulate behavior which should be shared among interacting objects and distributed to different parts of the software system. Roles are a dynamic classification concept for objects: They can be acquired and abandoned. Thus, role modeling has a highly dynamic semantics. For illustration, let us imagine an exemplary role model as shown in Fig. 1. The model consists of three base types (`Student`, `Professor` and `AssistantProfessor`), two role types (`Lecturer` and `Participant`) contained in a `Lecture`, a role-play relation, as well as some describing attributes. Such role-play relations are typically augmented by a number of constraints: A student can participate in multiple lectures, as well as a professor can hold several lectures – but never at the same time. While being student of different branches may be valid – even at the same time. Such additional role-play constraints will not be addressed in this paper, but in our future work. A possible way of applying FCA would be the usage of many-valued contexts to describe multiple role-play and detect changes and consistency via using a suitable scaling. An overview of possible role-play constraints can be found in [8].

To superimpose several role models in accordance to the role-play relation to a combined system the technique of **role composition** is used. This technique describes the merging of role and base types and is applied by introducing relations between these which describe the ability of role-play between these types. During system run-time the behavior of the base type instances is, thus, augmented by role behavior.

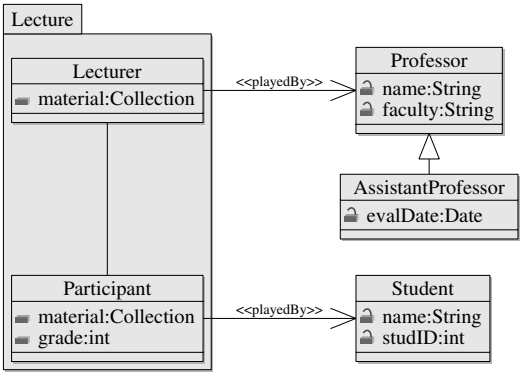


Fig. 1. Example role model *Lecture*

A formal representation for role models, as intended in this paper, needs to provide means for role model checking, e. g. for checking consistency of role models and role model composition. Furthermore, such a representation needs to cope with the dynamics of role models at run-time. Role modeling can benefit from the mathematic formalization, several knowledge deduction algorithms and the easy adaptability of Formal Concept Analysis. It has proven a value in lots of applications, e. g. in restructuring and refactoring software systems [9] or in class hierarchy design [3]. This work, as well as [10], were the main influences towards this paper. While [9] and [3] propose different styles of modeling software systems with the use of appropriate contexts, [10] created a basic language for representing role models in an order-theoretic way. Thus, in this paper we try to merge both approaches.

Formal Concept Analysis emerged as an approach to restructure Lattice Theory, that was introduced by Garrett Birkhoff in the early 1940s (e. g. [1]). The basic elements of Formal Concept Analysis are called **formal contexts**, which are triplets (G, M, I) consisting of a set G of **objects**, a set M of **attributes** and a relation $I \subseteq G \times M$, which describes the incidence of objects and attributes. This structure comes along with a very simple and intuitive representation: the cross table. From such tables one can now derive a set of concepts that are valid in the world represented by the formal context. The idea hereby is the following: a maximal set of objects whose elements have a maximal set of attributes in common, form – together with this maximal set of attributes – a **(formal) concept**. I. e. a formal concept is a pair (A, B), with $A \subseteq G, B \subseteq M, A' = B, B' = A$, where

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

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

A is then called **extent**, B is called **intent** of the concept (A, B).

To relate these concepts to each other, one introduces a partial order on the set of all concepts via

$$(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_2 \subseteq B_1)$$

and can hence construct a concept hierarchy or – due to its properties – a **concept lattice** representing relationships or commonalities between certain concepts. It shall be noted that for each object $g \in G$ exists a smallest concept γg having g in its extent. Dually, for each attribute $m \in M$ exists a largest concept μm having m in its intent [2].

3 Representing Role Models and Role Composition as Formal Contexts

In this paper, we create a representation of role models and, thus, start with giving a formal representation of the role model itself.

As we have seen in Sect. 2 a role model consists of a set R of role types, a set B of base types and a certain relation P between role and base types which is often referred to as role-play relation. Thus, we define a role model as follows:

Definition 1. *Let R be a set of **role types** and B a set of **base types**. Let further be $P \subseteq B \times R$ a relation between base and role types. The triplet $\mathbb{C} := (B, R, P)$ is then called **simple (formal) role model**.*

For the purpose of this paper simple role models are sufficient so in the following we omit the particle *simple*[1].

As one can see, a formal role model already turns out to be a formal context. With the objects being the base types of the role model, each concept describes the roles which can be played by a set of base types. An interesting field of future research will be the application of certain techniques of Formal Concept Analysis – e. g. attribute exploration – to gain knowledge about implicit relations between different roles that are not yet modeled or to check the model itself for correctness.

3.1 Static Modeling of Role Models and Role Model Composition

In role models, base classes, roles and their relations are used to specify parts of the software system. In our framework we are going to provide means to represent these elements using formal contexts. To reflect the characteristic property that role and base types belong to two parallel modeling hierarchies [10], we need to regard them as objects of two different formal contexts. Following the ideas of Godin and Valtchev in [3] it turns out to be useful to take the attributes of base or role types in the role model as attributes of the formal context representing base or role types respectively. This gives us the advantage that the class hierarchies

[1] This notion arose because in [6] a formal role model was defined as a quintuple (B, R_b, R_u, P, A) bearing some further information on the system which is not necessary for the scope of this paper. (cf. [6, pp. 29-30]).

of both role and base types are preserved in the concept lattice of the respective context.

When describing a role model composition, however, it is necessary to keep the information of both – base and role – contexts in one single context. Thus, we need to construct a suitable combination of these two contexts. To realise such composition we use the role-play relation as a kind of mapping between role and base types. So the attributes of the base types are added to the intent of the role type they play. As we will see later, this combination enables us to check for several properties of the whole system.

Definition 2. *Let $\mathbb{C} = (B, R, P)$ be a formal role model, let M_B, M_R be sets of attributes and $I_B \subseteq B \times M_B, I_R \subseteq R \times M_R$ relations that describe the incidence of types and attributes. The formal context $\mathbb{K}_\mathbb{C} := (G, M, \nabla)$ with*

$$G := \dot{B} \cup \dot{R}$$
$$M := \dot{M}_B \cup \dot{M}_R$$
$$\nabla := I_B \cup I_R \cup \Delta$$

where

$$\Delta := \left\{(r, m) \mid \exists\, b \in B : bPr \wedge bI_B m\right\} \subseteq R \times M_B$$

*is then called **static composition context** of $\mathbb{C}$.*

The dot-notation used is the same as proposed in [2, p. 38] for conceptual scaling and is meant to guarantee the disjointness of the corresponding sets. It can be seen as a mapping which replaces the set B by the set of tuples $\{b\} \times B$, where b is used as a literal pointing out that a type $t \in B$ is indeed a base type[2].

Example 2. Let us again consider the example from Fig. 1. If we construct the static composition context as given in Def. 2 we obtain the cross table and the corresponding concept lattice depicted in 2.

Interpreting the concepts in this lattice, we see that *AssistantProfessor* is a subconcept of *Professor*[3] (which is due to the subtype-relationship). Furthermore, *Participant* is a subconcept of *Student*, as well as *Lecturer* is a subconcept of *AssistantProfessor* (and hence *Professor*). As we see, a concept describing a role type is a subconcept of a concept describing a base type if the base type is in role-play relationship with the role type. In [6, p. 51] there is a proven theorem that states that there exists an equivalence between the subconcept relation and the role-play relation if the context of the base types is atomistic[4] and clarified[5].

[2] Accordingly, $\dot{R} := \{r\} \times R$, $\dot{M}_B := \{b\} \times M_b$ and $\dot{M}_R := \{r\} \times M_R$.

[3] Being exact, one would have to say, "We see that the concept belonging to the object *AssistantProfessor* is a subconcept of the concept belonging to the object *Professor*.". For improving the readability, we omit this lengthy notation and keep on using the – slightly inexact – short form.

[4] A formal context (G, M, I) is called **atomistic** if for each object $b \in B$ no object $\tilde{b} \in B$ exists with $\gamma\tilde{b} < \gamma b$.

[5] A formal context (G, M, I) is called **clarified** if $\gamma g = \gamma h \Rightarrow g = h$ for all $g, h \in G$ and dually for the attributes.

∇_{lec}	name	faculty	evalDate	studID	material	grade
Professor	×	×				
AssistantProfessor	×	×	×			
Student	×			×		
Lecturer	×	×	×		×	
Participant	×				×	×

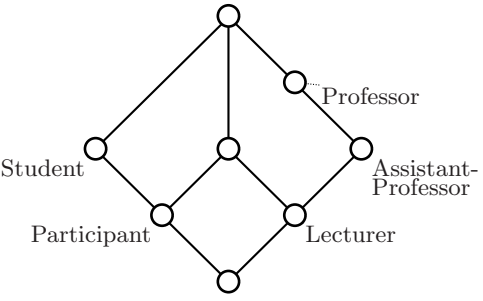


Fig. 2. Cross table and concept lattice of the static composition context of *Lecture*

We will present a slightly modified version of this theorem in Sect. 4 and state its impact on role modeling.

Concluding this section, we shall remark that the representation of a system $\{\mathbb{C}_k \mid k \in K\}$ of role models can be done likewise[6]. Each role model is then represented by an appropriate composition context (G_k, M_k, ∇_k). The whole system will then be modeled by joining the composition contexts in the following way

$$(G, M, \nabla) := \bigcup_{k \in K} (G_k, M_k, \nabla_k)$$

3.2 Representation of the Dynamic Run-Time State of a Role-Based Software System

Until now we have only modeled a static representation of role models and role model composition. That means that we have declared the role and base types which participate in the model and how they interact. However, for gaining knowledge about the lifecycle of instances of base types in our modeled system, we have to add the instance names to the extents of the respective concepts in the context describing the role composition. With such enriched contexts we are able to observe the lifecycle of each base type instance in a sequence of concept lattices.

The first step in this process is to encode the role-play relation directly in the set of concepts. Therefore it is necessary to extend the set of objects of the static composition context by each role-base-combination. The incidence relation of such a context should then be adapted so that each role-base-combination has all the attributes belonging to either of the types.

Definition 3. *Let* $\mathbb{C} = (B, R, P)$ *be formal role model and* $\mathbb{K}_{\mathbb{C}} = (G, M, \nabla)$ *the static composition context of* $\mathbb{C}$*. The formal context* $\tilde{\mathbb{K}}_{\mathbb{C}} := (\tilde{G}, \tilde{M}, \tilde{\nabla})$ *with*

$$\tilde{G} := \dot{B} \cup \dot{R} \cup P$$
$$\tilde{M} := M$$
$$\tilde{\nabla} := I_B \cup I_R \cup \tilde{\Delta}_B \cup \tilde{\Delta}_R$$

[6] K is an arbitrary set of indices.

where

$$\tilde{\Delta}_B := \big\{\big((b,r),m\big) \mid bI_Bm, b \in B\big\} \subseteq P \times M_B$$
$$\tilde{\Delta}_R := \big\{\big((b,r),m\big) \mid rI_Rm, r \in R\big\} \subseteq P \times M_R$$

is called ***template context*** *of* $\mathbb{C}$*.*

We expressed the extension from static composition contexts to template contexts by introducing a new concept $\gamma(b,r)$ in $\tilde{\mathbb{K}}_{\mathbb{C}}$ whenever γr is a subconcept of γb in $\mathbb{K}_{\mathbb{C}}$. This idea is illustrated in Fig. 3. The name "template context" is derived from the fact that the respective concept lattice can be seen as a template for the concept lattices of certain dynamic composition contexts. These dynamic composition contexts are introduced in Def. 4. For further clarification, see also Thm. 2.

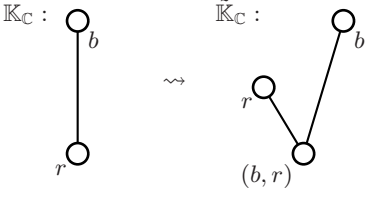


Fig. 3. The replacement schema which adds a concept for each role-base-pair in the template context

Let $T \subseteq \mathbb{N}$ describe the *run-time* of the system. Then, let $\mathcal{I}^t$ be the set of all active instances of the system at the moment $t \in T$. For each base type $b \in B$ may $\mathcal{I}_b^t$ describe the set of all instances of type b at the time $t \in T$. Analogously, $\mathcal{I}_r^t$ shall describe the active instances at the time $t \in T$ that play the role $r \in R$. Obviously the following holds

$$\bigcup_{b \in B} \mathcal{I}_b^t = \mathcal{I}^t$$

$$\bigcup_{r \in R} \mathcal{I}_r^t \subseteq \mathcal{I}^t$$

An instance $i \in \mathcal{I}_b^t \cap \mathcal{I}_r^t$ has access to all the attributes of its base type b and of the played role r. So it appears natural to define a dynamic composition context as follows:

Definition 4. *Let* $\mathbb{C} = (B, R, P)$ *be a formal role model,* $\tilde{\mathbb{K}}_{\mathbb{C}}$ *the proprietary template context and* $t \in T$ *a certain point of time. Let* $\mathcal{I}^t$ *be the set of all active instances at the point of time* t*,* $\mathcal{I}_b^t$ *be the set of all active instances of type* $b \in B$ *and* $\mathcal{I}_r^t$ *be the set of all active instances playing the role* $r \in R$*. Then, the context* $\tilde{\mathbb{K}}_{\mathbb{C}}^t := \big(\tilde{G}^t, \tilde{M}^t, \tilde{\nabla}^t\big)$ *with*

$$\tilde{G}^t := \dot{B} \cup \dot{R} \cup \mathcal{I}^t$$
$$\tilde{M}^t := \tilde{M}$$
$$\tilde{\nabla}^t := I_B \cup I_R \cup \tilde{\Delta}_B^t \cup \tilde{\Delta}_R^t$$

$\tilde{\nabla}_{\text{lec}}$	name	faculty	evalDate	studID	material	grade
Professor	×	×				
AssistantProfessor	×	×	×			
Student	×			×		
Lecturer					×	
Participant					×	×
ProfLecturer	×	×			×	
AssProfLecturer	×	×	×		×	
StudParticipant	×			×	×	×

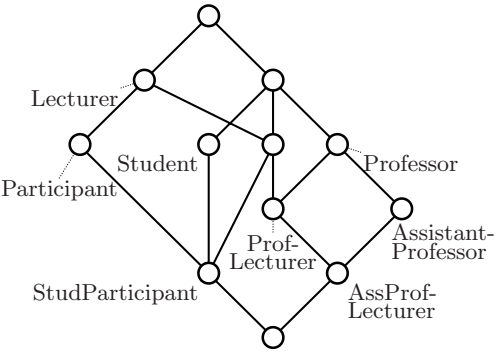


Fig. 4. Cross table and concept lattice of the template context of *Lecture*

is called ***dynamic composition context****. Thereby*

$$\tilde{\Delta}^t_B := \left\{(i,m) \mid \exists\ b \in B : i \in \mathcal{I}^t_b \wedge bI_Bm \text{ for } m \in M_B\right\}$$
$$\tilde{\Delta}^t_R := \left\{(i,m) \mid \exists\ r \in R : i \in \mathcal{I}^t_r \wedge rI_Rm \text{ for } m \in M_R\right\}$$

Example 3. Let us again consider the example role model in Fig. 1. The template context and its concept lattice is shown in Fig. 4.

Let us think of a professor `Aßmann` playing the role of the *Lecturer* and two students `Mühle` and `Wende` who participate in this lecture. We get as initial[7] sets of instances $\mathcal{I}^0 = \{\texttt{Aßmann}, \texttt{Mühle}, \texttt{Wende}\}$, $\mathcal{I}^0_{\texttt{Professor}} = \{\texttt{Aßmann}\}$, $\mathcal{I}^0_{\text{Participant}} = \{\texttt{Mühle}, \texttt{Wende}\}$ and so on. Thus, the initial concept lattice of the dynamic composition context is depicted in Fig. 5.

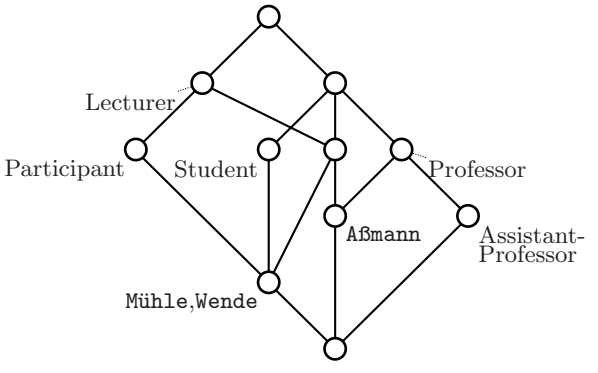


Fig. 5. Initial concept lattice of the dynamic composition context $\tilde{\mathbb{K}}^0_{\mathbb{C}}$

As we see the two lattices in Fig. 4 and Fig. 5 are pretty similar. We will discuss the relation between these two types of formal contexts in Sect. 4.2 and sketch how they can be used for visualisation and analysis.

[7] I. e. $t = 0$.

4 Contribution to Role Modeling

After introducing a formal representation of statics and dynamics for role models in terms of Formal Concept Analysis we will now discuss the contributions of our approach for role modeling.

4.1 Correctness of Role-Based System Specifications

Based on our static representation for role models and their superimposition in composition contexts we can now check the role-based system specifications for conflicts.

Let us first think about the example from Fig. 2. We notice that this context is not atomistic at all because *AssistantProfessor* is a subconcept of *Professor*. So the assumption of the theorem in [6, p. 51] is violated, the conclusion, however, holds. This motivates the easing of the assumptions as presented in Thm. 1. Consider the following modification of this example.

Example 4. When allowing the *AssistantProfessor* to play the *Participant* role as well, we receive the slightly modified role model in Fig. 6.

Constructing the concept lattice of the according composition context, we see that due to the subtype-relation between `Professor` and `AssistantProfessor`, *Professor* is now a super-concept of *Participant*, even though a `Participant` cannot be played by a `Professor`.

The difference between the role models in Fig. 1 and Fig. 6 is that in the modified example a subtype (`AssistantProfessor`) is playing a role (`Participant`), which its supertype (`Professor`) does not. This motivates the distinction between role models that allow this case and such that do not.

Definition 5. *Let $\mathbb{C} = (B, R, P)$ be a formal role model and let $(B, B, \leq_B)$ be a context describing the inheritance relationship on the base types. $\mathbb{C}$ is then called* ***extending*** *if there exist (at least) two base types $b_1, b_2 \in B$ with $b_1 \leq_B b_2$ and $\gamma b_1 < \gamma b_2$ (in $\underline{\mathfrak{B}}(B, R, P)$). $\mathbb{C}$ is called* ***non-extending*** *if it is not extending.*

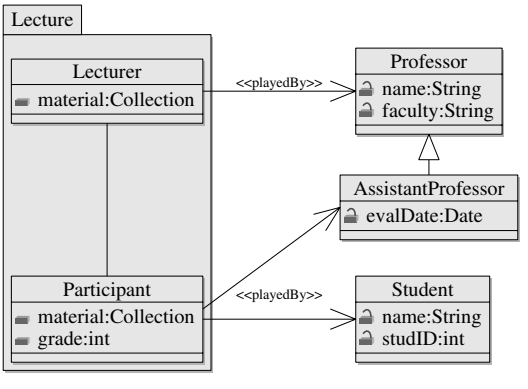


Fig. 6. Modified example role model *Lecture*

∇'_{lec}	name	faculty	evalDate	studID	material	grade
Professor	×	×				
AssistantProfessor	×	×	×			
Student	×			×		
Lecturer	×	×	×		×	
Participant	×	×	×	×	×	×

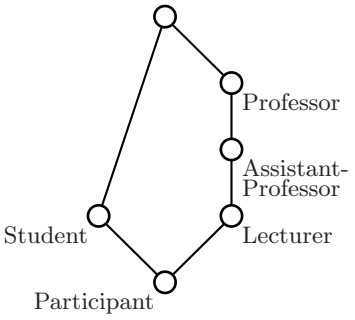


Fig. 7. Cross table and concept lattice of the modified role model from Fig. 6

As we have seen in Sect. 2 the role-play relation is inherited down the inheritance hierarchy of the base types. I. e. for two base types $b_1 \leq_B b_2$ the intents $\{b_1\}'$ and $\{b_2\}'$ of the concepts γb_1 resp. γb_2 are at least greater or equal. Adding a role r to $\{b_2\}'$ means that it is immediately inherited to $\{b_1\}'$. However, adding a role r to $\{b_1\}'$ does not affect $\{b_2\}'$. I. e. it holds $\{b_2\}' \subset \{b_1\}'$ which means $\gamma b_1 < \gamma b_2$.
So, whenever one finds two base types b_1 and b_2 with $b_1 \leq_B b_2$ and $\gamma b_1 < \gamma b_2$ one knows that there exists at least one role type $r \in R$ that can be played by b_1 but not by b_2. Thus b_1 extends the role-play of b_2.

With this definition, we can extend the theorem from [6, p. 51]:

Theorem 1. *Let $\mathbb{C} = (B, R, P)$ be a non-extending role model and $\mathbb{K}_{\mathbb{C}} = (G, M, \nabla)$ the composition context to $\mathbb{C}$. Let the context (B, M_B, I_B) of base types further be clarified. Then for all $b \in B, r \in R$ holds*

$$\gamma r \leq \gamma b \Leftrightarrow bPr$$

Proof. "⇐": Clear by the definition of ∇.
"⇒": Be $b \in B, r \in R$, with $\gamma r \leq \gamma b$, thus $\{b\}' \subseteq \{r\}'$. Assume that $(b, r) \notin P$. By definition of ∇ there have to exist base types b_n for each $n \in \{b\}'$, with $b_n Pr$. This means that

$$\{b\}' = \bigcap_{n \in \{b\}'} \{b_n\}'$$

which leads with the basic theorem of Formal Concept Analysis [2, p. 20] to

$$\gamma b = \bigvee_{n \in \{b\}'} \gamma b_n$$

Thus, $\gamma b_n \leq \gamma b$ for each $n \in \{b\}'$. This can, again by the construction of (G, M, ∇), only be true if each of the b_n describe a subtype of the base type b. Since $\mathbb{C}$ is non-extending, it has to be $\gamma b_n = \gamma b$. Since (B, M_B, I_B) is clarified, $b_n = b$ which leads to a contradiction having $b_n Pr$ and $(b, r) \notin P$ at the same time. □

This theorem can now be used to check the role model for correctness. Whenever the corresponding composition context is *specifying*, the role-play relation is not modeled properly. This does not only allow the detection of conflicts at modeling time but also helps to estimate the impact of changes during the evolution of role-based system specifications.

4.2 Visualisation and Analysis of System Dynamics at Run-Time

In the same way as Steimann's modeling language LODWICK is able to present *snapshots* of the system (i. e. it delivers all necessary information on the system, e. g. the union of all active instances, all base and role types and the according role-play), our approach does. But in contrast to LODWICK which represents these snapshots only as sets, we are able to present concept lattices, covering the whole system at the certain point of time. The visualisation via concept lattices gives an intuitive, clearly arranged and comprehensive diagrammatic schema of the relations between the system's objects. However, it is hardly possible to compute the concept lattice of each dynamic composition context for each point of time. The concept lattice is rather seen as a means to visualize parts of the system on-demand, as an above-mentioned snapshot. Additionally, one would not likely demand to see the whole lattice but only the relevant parts of it, e. g. the order ideal[8] of a certain concept containing all active instances of the according type. Future work will show how such visualisation mechanisms will be applied.

Another benefit of the presented modeling approach is the following: When we compare the lattices Fig. 4 and Fig. 5 we see that the concept lattice of the dynamic composition context can be embedded into the concept lattice of its template context. The following theorem stems from [6] and describes the relation of these two lattices in more detail.

Theorem 2. *Let $\mathbb{C} = (B, R, P)$ be a formal role model, $\tilde{\mathbb{K}}_{\mathbb{C}}$ the proprietary template context and $\tilde{\mathbb{K}}^t_{\mathbb{C}}$ a dynamic composition context at the point of time $t \in T$. Then, $\underline{\mathfrak{B}}\big(\tilde{\mathbb{K}}^t_{\mathbb{C}}\big)$ can be embedded order- and (even more) join-preserving in $\underline{\mathfrak{B}}\big(\tilde{\mathbb{K}}_{\mathbb{C}}\big)$.*

So, already at role modeling time, we can give an upper bound for the cardinality of the set of concepts that are valid. There will be no point of the system's run-time, when more concepts will be necessary to form the concept lattice of the dynamic composition context than allowed by the number of concepts of the according template lattice. This helps to estimate at modeling time how much space is required for representing the dynamic composition context during run-time.

As role modeling was intended to handle dynamic and shared behavior, it is natural that role types adapt and change certain behavior of base types. This can be reached by redefining methods of the base type in playable role types. In our approach, we defined a formal context by using type attributes only. Godin

[8] In an ordered set $(V, \leq)$, an order ideal $(a]$ for $a \in V$ is defined as $(a] := \{v \in V \mid v \leq a\}$.

and Valtchev have, however, shown how to create formal contexts preserving the type hierarchy using type methods [3, p. 9]. With adapting our approach towards type methods, we gain further benefits, as sketched below.

Example 5. Assume a base type `Professor` implementing a method `scribbleOnBoard()`. How the actual scribbling is performed depends on the role this professor is playing. As a `Lecturer` – maybe under time pressure – the scribbling would probably be less neat than usually. In an appropriate context describing the dynamic role model using type methods the according concepts' intents would then share an attribute *scribbleOnBoard*. If an instance of the base type `Professor` lies in the order ideal of the concept *Lecturer* it follows that the `Professor`'s method `scribbleOnBoard` has to be dispatched by the according method of `Lecturer`.

A more detailed elaboration of this approach, as well as establishing further theorems offer lot of material for future research.

5 Conclusion

In this paper we contributed a formal representation for role models and the statics and dynamics of role model composition. This representation is directly applicable in a computational framework for analysing role-based systems and their specifications. We introduced fundamental theorems for checking role-based system specifications for conflicts and to represent the run-time state of a role-based system using dynamic composition contexts. These composition contexts were applied for visualising the run-time state and, thus, for modeling the life-cycle of instances of base types. We are thus able to identify role changes and type polymorphism.

Another task that is to be done is the extension of our attribute-based approach towards a method-based one. As sketched in the last section such an approach will help to identify role polymorphism and method dispatch.

To support role-based system engineering practically, future work will have to provide an implementation and evaluation of the feasibility of the presented approach in an integrated software modeling and analysis tool. This also includes tooling to realise the checking and visualisation proposed in this paper.

As we stated earlier, there are open fields in exploiting the context structure of the role model itself as well as applying context constructions for role-play constraints.

Acknowledgments

We would like to thank Uwe Aßmann and Bernhard Ganter who supported this work with their valuable suggestions and fruitful discussions. This work was partially funded by the EC in the FP7 project MOST.

References

1. Birkhoff, G.: Lattice Theory. Colloquium Publications, vol. 25. American Mathematical Society (1940)
2. Ganter, B., Wille, R.: Formale Begriffsanalyse: Mathematische Grundlagen. Springer, Heidelberg (1996)
3. Godin, R., Valtchev, P.: Formal Concept Analysis-Based Class Hierarchy Design in Object-Oriented Software Development. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 304–323. Springer, Heidelberg (2005)
4. Herrmann, S.: A Precise Model for Contextual Roles: The Programming Language ObjectTeams/Java. Applied Ontology 2(2), 181–207 (2007)
5. Kristensen, B.B., Østerbye, K.: Roles: Conceptual Abstraction Theory and Practical Language Issues. Theory and Practice of Object Systems 2(3), 143–160 (1996)
6. Mühle, H.: Modellierung rollenbehafteter Typhierarchien unter Verwendung der Formalen Begriffsanalyse. Diplomarbeit, Technische Universität Dresden (2009)
7. Reenskaug, T., Wold, P., Lehne, O.A.: Working with Objects: The Ooram Software Engineering Method. Manning Publications (1996)
8. Riehle, D., Gross, T.: Role Model-based Framework Design and Integration, pp. 117–133. ACM Press, New York (1998)
9. Snelting, G., Tip, F.: Understanding Class Hierarchies using Concept Analysis. ACM Trans. Program. Lang. Syst. 22(3), 540–582 (2000)
10. Steimann, F.: On the Representation of Roles in object-oriented and conceptual Modelling. Data Knowledge Engineering 35(1), 83–106 (2000)

Approaches to the Selection of Relevant Concepts in the Case of Noisy Data

Mikhail Klimushkin[1], Sergei Obiedkov[1], and Camille Roth[2]

[1] Higher School of Economics, Moscow, Russia
klim.mikhail@gmail.com, sergei.obj@gmail.com
[2] CAMS (CNRS/EHESS), Centre d'Analyse et de Mathématique Sociales
EHESS, 54 Bd Raspail, F-75006 Paris, France
roth@ehess.fr

Abstract. Concept lattices built on noisy data tend to be large and hence hard to interpret. We introduce several measures that can be used in selecting relevant concepts and discuss how they can be combined together. We study their performance in a series of experiments.

1 Introduction

Formal Concept Analysis (FCA) as a categorization method aims at grouping objects described by common attributes. In this framework, a category is more precisely defined as a maximal set of objects sharing a maximal set of attributes. Such groupings are then gathered in a hierarchical, lattice-based structure which straightforwardly exhibits various relationships between categories and their sub- and super-categories. As such, this approach provides an ideal formalization of categories in terms of concepts as they are traditionally defined philosophically, i.e., concepts extensionally described by sets of entities and intensionally described by attribute sets.

The taxonomical structure of a given domain is therefore frequently expected to be naturally revealed by applying FCA to an empirical description of its objects and their attributes. However, in spite of these strong theoretical foundations, the translation of empirical data into clean and relatively readable structures remains a common issue. Indeed, FCA induces a potentially dreadful combinatorial complexity and the structures obtained even from small-sized datasets can become prohibitively huge. In this respect, noise constitutes a primary factor of complication as it favors the existence of many similar but distinct concepts, which may excessively inflate the lattice with superfluous information to the cost of significantly impaired readability.

Hence, displaying interesting patterns while removing useless and cumbersome information constitutes a crucial task when assuming a noisy dataset. Such patterns are admittedly those which are or would be the only ones to be found in an ideally clean (non-noisy) dataset. This issue has seemingly received little attention in the FCA literature, apart from methods targeted at simplifying lattices [1,2,5,6,10]. In this paper, we design noise filtering criteria to specifically

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 255–266, 2010.

account for the likeliness of a concept to exist because of noise rather than to reflect essential features of the underlying taxonomy. To proceed, we essentially aim at appraising the diverse efficacy of such indices on basic datasets altered by simple noise effects.

The paper is organized as follows: in Sect. 2, we recall the principles and notations of FCA. Section 3 introduces the various indices and their rationale, while Sect. 4 describes their application on noisy contexts and comments the corresponding results.

2 FCA Definitions and Related Work

Before proceeding, we briefly recall the FCA terminology [3]. Given a *(formal) context* $\mathbb{K} = (G, M, I)$, where G is called a set of *objects*, M is called a set of *attributes*, and the binary relation $I \subseteq G \times M$ specifies which objects have which attributes, the derivation operators $(\cdot)^I$ are defined for $A \subseteq G$ and $B \subseteq M$ as follows:

$$A^I = \{m \in M \mid \forall g \in A : gIm\};$$
$$B^I = \{g \in G \mid \forall m \in B : gIm\}.$$

In words, A^I is the set of attributes common to all objects of A and B^I is the set of objects sharing all attributes of B.

If this does not result in ambiguity, $(\cdot)'$ is used instead of $(\cdot)^I$. The double application of $(\cdot)'$ is a closure operator, i.e., $(\cdot)''$ is extensive, idempotent, and monotonous. Therefore, sets A'' and B'' are said to be *closed*.

A *(formal) concept* of the context (G, M, I) is a pair (A, B), where $A \subseteq G$, $B \subseteq M$, $A = B'$, and $B = A'$. In this case, we also have $A = A''$ and $B = B''$. The set A is called the *extent* and B is called the *intent* of the concept (A, B).

A concept (A, B) is a *subconcept* of (C, D) if $A \subseteq C$ (equivalently, $D \subseteq B$). In this case, (C, D) is called a *superconcept* of (A, B). We write $(A, B) \leq (C, D)$ and define the relations $\geq$, $<$, and $>$ as usual. If $(A, B) < (C, D)$ and there is no (E, F) such that $(A, B) < (E, F) < (C, D)$, then (A, B) is a *lower neighbor* of (C, D) and (C, D) is an *upper neighbor* of (A, B); notation: $(A, B) \prec (C, D)$ and $(C, D) \succ (A, B)$.

The set of all concepts ordered by $\leq$ forms a lattice, which is denoted by $\underline{\mathfrak{B}}(\mathbb{K})$ and called the *concept lattice* of the context $\mathbb{K}$. The relation $\prec$ defines edges in the *covering graph* of $\underline{\mathfrak{B}}(\mathbb{K})$. The meet and join in the lattice are denoted by $\wedge$ and $\vee$, respectively.

3 Indices for Concept Selection

3.1 Stability

The stability index describes the proportion of subsets of objects of a given concept whose closure is equal to the intent of this concept [6]. In other words, it is meant to capture how much a concept intent depends on particular objects

of the extent: should some objects be removed from the concept extent, would the concept intent remain the same? In an extensional formulation, the index specifies how much a concept extent depends on intent attributes.

We thus distinguish between intensional and extensional stability indices σ_i and σ_e:

$$\sigma_i(A, B) = \frac{|\{C \subseteq A \mid C' = B\}|}{2^{|A|}}$$

$$\sigma_e(A, B) = \frac{|\{D \subseteq B \mid D' = A\}|}{2^{|B|}}$$

The intent of a concept with a high intensional stability index would be likely to exist even if we ignore several of its objects: it does not disappear if the intent of some of its objects is modified, e.g., if these objects lose some of the properties of this concept. Put differently, concepts relying on noisy objects and, therefore, not typical of a realistic category, are more likely to be unstable. Similarly, extensional stability helps isolating concepts that appear because of noisy attributes.

Given the covering graph of a concept lattice, computing stability for all concepts can be done using the algorithm presented in [9]. This algorithm is essentially quadratic in the number of concepts in the lattice, which may be prohibitively expensive for large lattices. On the other hand, this algorithm needs a lattice as input, and generation of lattices for very large datasets may be impractical, anyway. Hence, it would be useful to develop an algorithm that generates only stable concepts directly from the input context (perhaps, giving up on the exact computation of stability and computing only approximate estimates).

Below, we mainly rely on intensional stability, which we denote by σ for the sake of clarity.

3.2 Concept Probability

A concept that covers fewer objects is normally less intensionally stable than a concept covering more objects. Still, these rather specific concepts can correspond to interesting associations that should not be ignored in the analysis. To give such specific concepts a chance of surviving the stability test, we need to normalize the stability index. To this end, we introduce the notion of *concept probability*. The idea is that if a concept has low probability, but is still observed in the data, it may reflect an interesting dependency and should be taken into account.

The relation between the presence of some patterns and some simple features of 01-matrices has received some attention in the literature, mainly by appraising how some patterns could be artifactual with respect to a particular *a priori* knowledge on the structure of such matrices. These studies focus in particular on the effect of matrix marginals—i.e., the distributions of sums of rows and columns—on patterns. For instance, the so-called configuration model of [8] assumes a null-model of random matrices conserving only original marginals; then, estimates the probability of some statistical features, generally related to

the topology of the matrix interpreted as the adjacency matrix of a graph. Closer to FCA, [4] later proposed to estimate whether frequent itemsets could be due to chance by, again, relating their presence to marginals: patterns are subsequently said to be artifactual if they are found in comparable proportions in the original data and its randomized version.

The notion of concept probability essentially follows a relatively similar line of reasoning. For $m \in M$, denote by p_m the probability of an arbitrary object having the attribute m. For $B \subseteq M$, define p_B, the probability of an arbitrary object having all attributes from B, by

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

thus assuming the mutual independence of attributes. By denoting $n = |G|$, we obtain the following formula for the probability of B being closed:

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

To see the reasoning behind the formula, note that, for $|B'| = k$ and $B = B''$, we need that

1. There are k objects that have all attributes from B;
2. Each of the other $n - k$ objects does not have at least one attribute from B;
3. No attribute outside B belongs to all the k objects.

There are $\binom{n}{k}$ variants to choose k objects. The probability that each of the chosen k objects has all attributes from B is p_B^k, and the probability that each of the other $n-k$ objects does not have at least one attribute from B is $(1-p_B)^{n-k}$. The probability that not all of the k chosen objects have an attribute m is $(1 - p_m^k)$; hence the probability that none of the attributes outside B belongs to all the k objects is $\prod_{m \notin B}(1 - p_m^k)$. Therefore, the joint probability of $|B'| = k$ and $B = B''$ is

$$\binom{n}{k} p_B^k (1 - p_B)^{n-k} \prod_{m \notin B} (1 - p_m^k).$$

By summing over k, we obtain the complete formula for the probability of an attribute set B being closed, which is given above.

Again, one can regard this as "intensional probability" of a concept and define "extensional probability" dually (as the probability of an object subset being closed).

It can be easily seen that it is possible to compute $\binom{n}{k+1} p_B^{k+1} (1 - p_B)^{n-(k+1)}$ from $\binom{n}{k} p_B^k (1 - p_B)^{n-k}$ using a constant number of multiplication operations, while $\prod_{m \notin B}(1 - p_m^k)$ requires $O(|G||M|)$ multiplications. Thus, computing the (intensional) probability of one concept involves $O(|G|^2|M|)$ multiplication operations.

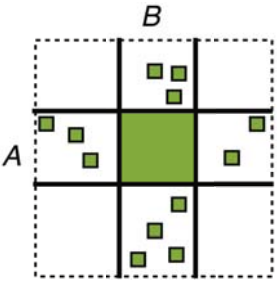


Fig. 1. Separation index $\mathfrak{s}(A, B)$ corresponds to the ratio between the green central area $|A| \times |B|$ and the total area covered by intents of every object of A and extents of every attribute of B

3.3 Separation

The separation index is meant to describe how well a concept sorts out the objects it covers from other objects and, jointly, how well it sorts out the attributes it covers from other attributes of the context.[1]

Put differently, it indicates the significance of the difference between the objects covered by a given concept from other objects and, at the same time, between its attributes and other attributes.

To do so, the separation index $\mathfrak{s}$ is defined as the ratio between the area covered in the context by a concept (A, B) and the total area covered by its objects and attributes (see Fig. 1 for an illustration):

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

Obviously, $\mathfrak{s}(A, B)$ can be computed in $O(|G| + |M|)$ time (assuming that object intents and attribute extents are pre-computed).

4 Reconstruction of Noisy Datasets

4.1 Noisy Contexts

We understand noise in all generality as a measurement discrepancy between an empirical (real) setting and what a given dataset says about it. Empirical data may be noisy for various reasons: it can be due for instance to a lack of precision in collecting or building the dataset by mistakenly adding extra attributes to some objects or omitting some objects in describing the extent covered by some attribute. Noise can also be understood as exceptions to a rule, when attempting to exhibit clear-cut joint object-attribute categories, i.e., noisy objects or attributes. We therefore distinguish two different types of noise:

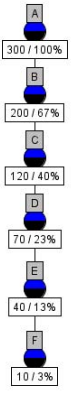
[1] A similar motivation is behind "relevant bi-sets" described in [2], but our approach is different.

(Type I) — either by altering every cell value in the context with some probability;
(Type II) — or by adding to the original context a given number or proportion of completely random objects or attributes.

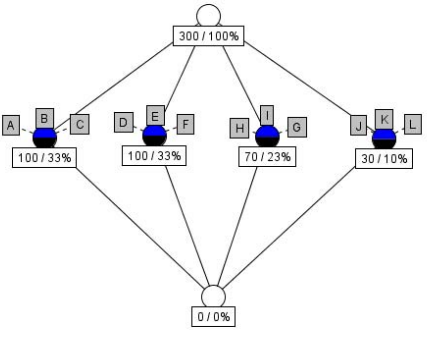
4.2 Example Contexts

We used four simple contexts with rather basic structures. This includes a chain-based lattice (300 objects, 6 attributes), an antichain-based lattice (300 objects, 12 attributes), and two more elaborate structures respectively built upon 300 and 400 objects and 6 and 4 attributes. The corresponding lattices are represented in Fig. 2.

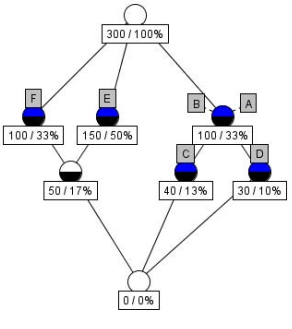
Every concept of these lattices contains many identical objects, often characterized by only a handful of attributes, sometimes, only one attribute.



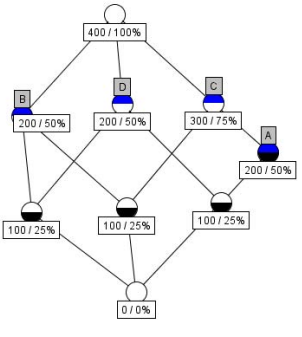
Context 1 (chain)



Context 2 (anti-chain)



Context 3



Context 4

Fig. 2. Original contexts

Context 1, With Noise

type I			type II
separation $\mathfrak{s}$	stability σ		
2%	10%	20%	

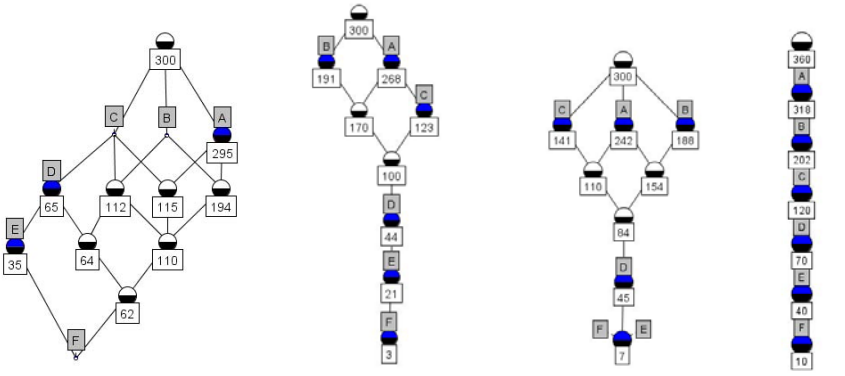


Fig. 3. Noisy instances of context 1 (chain lattice) with various filtering strategies. *Three first lattices, from left to right:* type-I noise using respectively 2, 10 and 20% noise, together with separation (left) or stability (middle lattices). The ideal result (not shown in the picture) is achieved by the combination of stability and probability $\sigma(A,B) - k \cdot p(B = B'')$. *Rightmost lattice:* type-II noise using 20% object addition and stability-based pruning.

In each of these contexts, we uniformly introduce noise (type I or type II), which leads to the appearance of many new concepts, and then try to find the concepts of the original context among the concepts of the noisy context using various combinations of the indices discussed above.

4.3 Results

Context 1—Chain Lattice. Stability is relatively successful at dealing with type I-noise, with only a few discrepancies with the original lattice and up to 20% of perturbation (correctly selecting the original six intents among 56 intents of the noisy lattice)—see Fig. 3. On the contrary, separation indices are much less effective, even with only 2% of noise.

The best result (not shown in Fig. 3) is achieved with the combination of intensional stability and probability $\sigma(A,B) - k \cdot p(B = B'')$. With appropriately chosen value of $k \geq 0$, it makes it possible to completely restore the original structure even with 20% of noise (in this case, we had $k = 0.0005$).

The relevance of stability is confirmed when examining contexts altered with type II-noise, both when random objects or random attributes are introduced (using intensional stability in the former case and extensional stability in the latter case). This should not come as a surprise, since intensional (extensional)

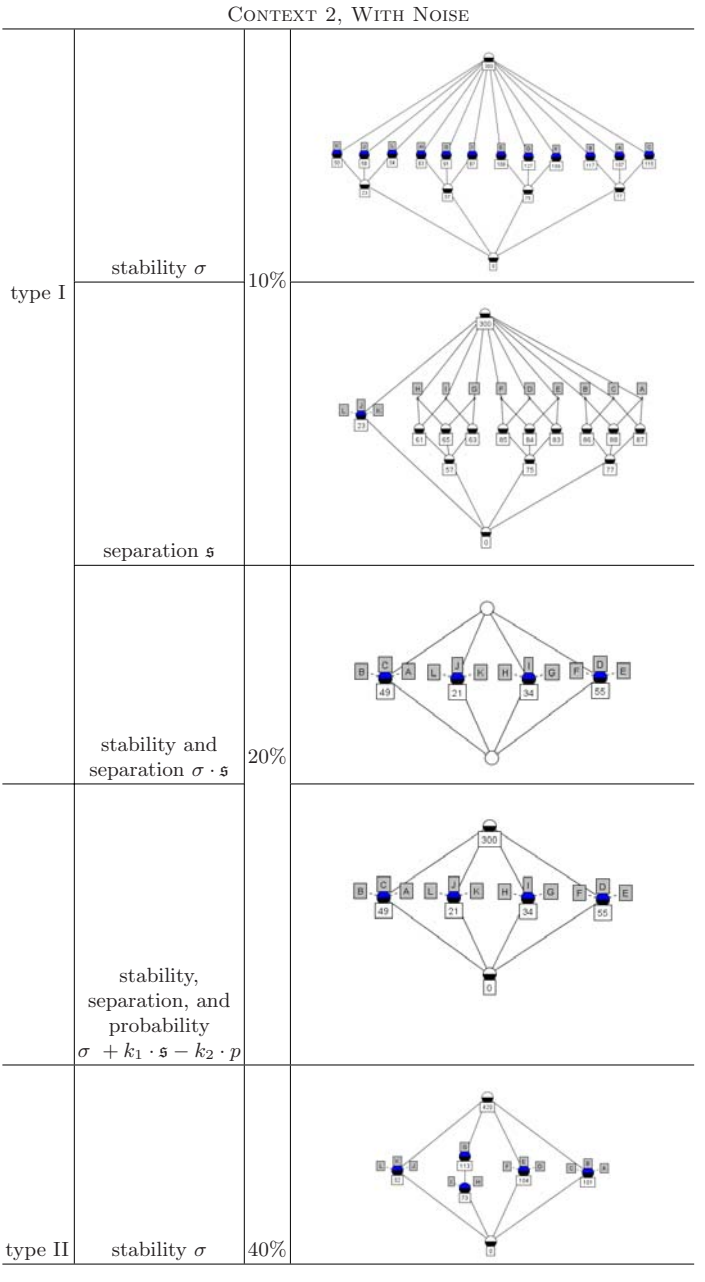


Fig. 4. Noisy instances of context 2 (antichain lattice) with various filtering strategies. *Four top lattices, from top to bottom:* type I noise using respectively 10, 10, 20, and 20% noise; together with stability, separation, the product of stability and separation, and the sum of weighted stability, separation, and probability. *Bottommost lattice:* type II-noise using 40% object addition and stability-based pruning.

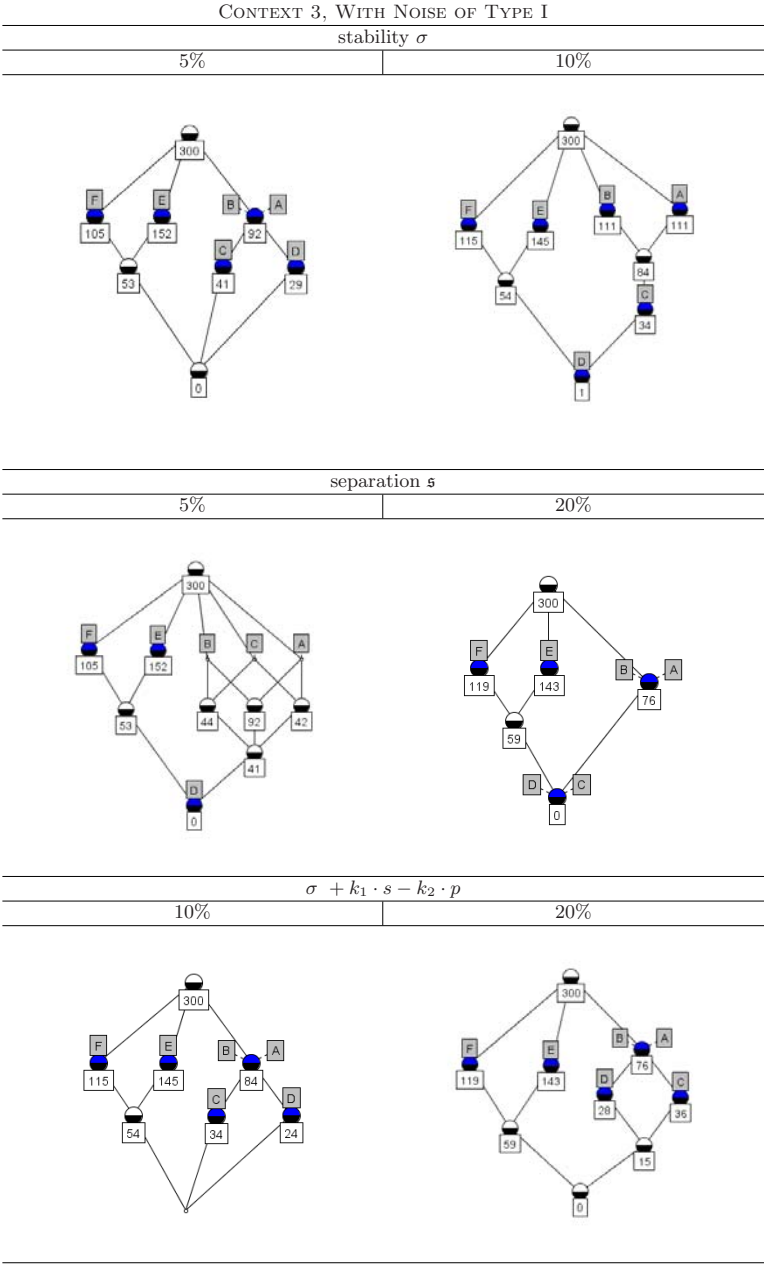


Fig. 5. Instances of context 3 with type I-noise and various filtering strategies. *Top lattices:* filtering lattices with contexts at 5% and 10% noise, using stability. *Middle lattices:* filtering with separation with 5% and 20% noise level. *Bottom lattices:* filtering with a combination of the three criteria with 10% and 20% noise level.

Context 4, With Noise of Type I

stability σ	separation $\mathfrak{s}$
10%	

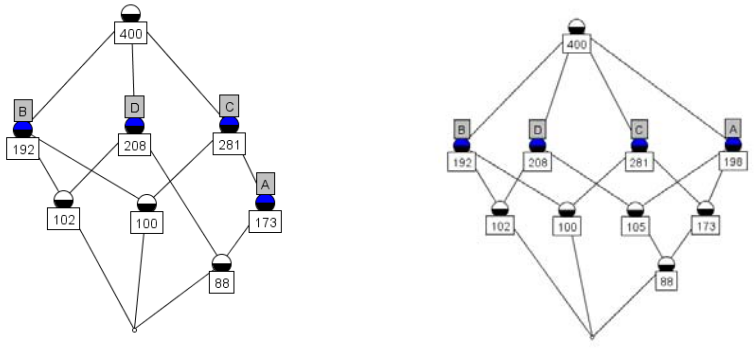


Fig. 6. Instances of context 4 with type I-noise at 10%, filtering with stability (*left*) or separation (*right*)

stability is defined specifically to address the case of noisy objects (attributes) added to the otherwise clean context (that is, exactly type II-noise).

Context 2—Antichain Lattice. As regards type-I noise, when the noise is relatively limited (10%, which yields 324 concepts), separation is quite equivalent to stability: separation produces more faithful lattices, while stability yields more readable structures. When using thresholds on both stability and separation, the lattice exhibits exactly the same structure as the original. At 20% of noise (786 concepts), this does not work anymore. Yet, stability normalized by probability remarkably yields the original structure again (see Fig. 4).

As regards type-II noise, as with the chain lattice case, stability is robust, even with 40% of noise (268 concepts in the noisy lattice).

Context 3. This context features several concepts which are either super- or sub-categories in distinct parts of the lattice. Context 3 therefore significantly diverges from the previous prototypical cases.

As regards type I-noise, stability performs relatively well: it perfectly yields the original structure at 5% noise (36 concepts) and remains close to this benchmark at 10% (52 concepts) and even 20% (63 concepts) noise (see Fig. 5, top).

On the other hand, separation yields comparatively unconvincing results, from just 5% of noise (see Fig. 5, middle). Combining stability with separation and probability allows us to achieve better results (see Fig. 5, bottom).

As usual, stability works just fine for type II-noise, even with 40% of noise.

Context 4. This lattice eventually offers a mix of sub- and super-concepts intertwined in a somewhat balanced manner. Stability, again, is the only criterion that yields the exact original structure. Separation, when used alone or in various combinations with stability, produces a slightly different structure, as shown in Fig. 6 (this applies, e.g., when using the sum or the product of σ and $\mathfrak{s}$).

Again, type II noise can be perfectly filtered out by stability alone.

4.4 Conclusion

On the whole, stability is remarkably effective at sorting out type II-noise. Since the most stable concepts are usually those which are the least affected by individual objects or attributes, the success of this criterion should be relatively unsurprising. Still, stability proves to be significantly helpful for type I-noise as well. Unlike type II-noise, type I-noise might make some original concepts disappear (in which case, none of the methods described in the paper is sufficient to reconstruct such concepts). However, the probability for a stable concept to disappear or become significantly less stable because of type I-noise is quite low, especially if the noise is relatively limited.[2] Additional experiments could help characterize this probability more precisely, thus explaining why stability behaves well even for type I-noise.

Separation is founded on a different rationale and, therefore, can be used only as an auxiliary criterion when dealing with noise. A notable exception is given by anti-chain-like parts of lattices, which can be efficiently reconstructed with separation. In other cases, it is less effective than stability. One of its shortcomings is that concepts with similar extents and intents tend to have similar separation indices. As a result, separation-based filtering prunes groups of similar concepts (Fig. 4). It seems promising to combine separation and stability, as they reduce each other's drawbacks.

Similarly, probability is not a self-sufficient criterion for filtering noise. Low or high probability of a concept does not say much of its importance. It can only correct other indices, serving a normalizing measure for stability or separation. It seems that the most promising combination is $\sigma + k_1 \cdot \mathfrak{s} - k_2 \cdot p$, where $k_2 < 0.2$. Such criterion seems able to reconstruct the original structure where stability and separation alone are not sufficient. Learning appropriate values for k_1 and k_2 can be done separately for different datasets. It remains to study the best ways to automatically learn these coefficients.

A next step would consist in validating these approaches on empirical data and theoretical interpretation of the performance of various combinations of the indices on different structures. It would also be interesting to see how the indices behave when noise is introduced in a non-uniform way. Automated learning of the best performing combinations (e.g., with evolutionary programming) is another research direction. This would most likely require defining a distance between two lattices, so as to be able to precisely evaluate the quality of reconstructing original lattices.

[2] We are grateful to the anonymous reviewer for this remark.

Acknowledgements

The first two authors were supported by the Scientific Foundation of the State University–Higher School of Economics (projects 08-04-0022 and 10-04-0017). The second author was also supported by the Russian Foundation for Basic Research (project 08-07-92497-NTsNIL_a). Line diagrams were produced with Concept Explorer (http://conexp.sourceforge.net/), a software tool developed by Serhiy Yevtushenko.

References

1. Bayardo Jr., R., Goethals, B., Zaki, M. (eds.): Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2004). CEUR-WS.org (2004)
2. Boulicaut, J.F., Besson, J.: Actionability and formal concepts: A data mining perspective. In: Medina, Obiedkov [7], pp. 14–31
3. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
4. Gionis, A., Mannila, H., Mielikäinen, T., Tsaparas, P.: Assessing data mining results via swap randomization. ACM Trans. Knowl. Discov. Data 1(3), 14 (2007)
5. Jay, N., Kohler, F., Napoli, A.: Analysis of social communities with iceberg and stability-based concept lattices. In: Medina, Obiedkov [7], pp. 258–272
6. Kuznetsov, S., Obiedkov, S., Roth, C.: Reducing the representation complexity of lattice-based taxonomies. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 241–254. Springer, Heidelberg (2007)
7. Medina, R., Obiedkov, S. (eds.): ICFCA 2008. LNCS (LNAI), vol. 4933. Springer, Heidelberg (2008)
8. Newman, M.E.J., Strogatz, S., Watts, D.: Random graphs with arbitrary degree distributions and their applications. Physical Review E 64(026118) (2001)
9. Roth, C., Obiedkov, S., Kourie, D.G.: Towards concise representation for taxonomies of epistemic communities. In: Yahia, S.B., Nguifo, E.M., Belohlavek, R. (eds.) CLA 2006. LNCS (LNAI), vol. 4923, pp. 240–255. Springer, Heidelberg (2008)
10. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with TITANIC. Data & Knowledge Engineering 42, 189–222 (2002)

Concept Analysis as a Framework for Mining Functional Features from Legacy Code

Amal El Kharraz, Petko Valtchev, and Hafedh Mili

Dépt. d'Informatique UQAM, C.P. 8888,
Succ. Centre-Ville, Montréal H3C 3P8, Canada
ekharraz@iro.umontreal.ca, valtchev.petko@uqam.ca, mili.hafedh@uqam.ca

Abstract. Legacy OO applications typically implement a set of functional features which, in the absence of aspect-oriented techniques to separately develop and maintain them, end up embodied in the same class hierarchies. We identified three types of design techniques used to implement that embodiment: a) multiple inheritance– or simulations thereof, b) aggregation/delegation, and c) what we referred to as *ad-hoc implementation*. We are interested in identifying and isolating software artifacts that implement distinct functional features. Here, we explore the use of concept analysis to detect ad-hoc implementations of functional features. We present the principles underlying our overall approach, a formalization of the problem in terms of concept analysis, a method for identifying functional features based on the construction and exploration of the concept latice, and the results of an experimental validation study.

1 Introduction

Legacy software, i.e. software whose development predates the latest trends in software engineering, typically implements a collection of functional features, identified either at requirement or at maintenance steps. For instance, assume a legacy information system where the same `Employee` class is used by both payroll and production planning functionalities. The former requires the support of data and function members pertaining to salary base, number of hours worked, etc., whereas the latter would emphasize on representing skills, shifts, number of hours worked, etc. We are interested in developing techniques for identifying functional features in legacy OO code for cases where several features affect the same classes. Minimally, this should help us understand the mechanics of a legacy application and maintain a given feature without interfering with the other ones. This should also be helpful whenever a refactoring of the application is to be performed to repackage features in a way so that they can be (further) developed and maintained separately, and intertwined on demand.

Ideally, our abstraction and packaging technique should allow us to realize different functional features in distinct software artifacts that we can develop, maintain, and compose at will [1]. The techniques provided by the OO paradigm enable the separation of *object-specific* features – by encapsulating them in classes

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 267–282, 2010.

– but do not adequately separate features that affect or crosscut a distinct subset of the application classes. Aspect-oriented software development (AOSD) was meant to address this problem by proposing artifacts that help untangle requirement types that OO abstractions could not. However, absent such techniques, designers have to cope with multiple functional features by means of conventional OO language constructs or design idioms. For instance, they may rely on multiple inheritance (either built-in or itself emulated through design tricks), delegation, or, alternatively, decide to use an ad-hoc implementation with no specific discipline for packaging of the functional features. In order to identify functional features, our overall strategy is to automatically detect such programming and design idioms within the legacy code. In this paper, we focus on ad-hoc implementations of multiple functional features and discuss a *formal concept analysis*-based approach for the detection of instances thereof.

In the remainder of the paper, section 2 discusses the concept of *functional feature* and provides a working definition thereof to guide a critical examination of the problem of identifying functional features in legacy code. Section 3 narrows the focus on *ad-hoc implementations* of multiple functional features and the detection of instances thereof. Section 4 presents a dedicated feature mining method while the results of an experimental study are discussed in section 5. Related work is summarized in section 6.

2 Characterizing Functional Features

We understand a *functional feature* as a slice or "subset" of an OO application that addresses a cohesive subset of its functional requirements. A functional requirement, as opposed to a *non-functional* one, deals with the ⟨input,output⟩ relationship implemented by the software. The non-functional part of the software requirements would specify the various conditions imposed on the production of the output. It has been observed that the elements of an OO application contributing to a functional feature will exhibit stronger "cohesion" than the elements of the application as a whole[2]. Going back to our `Employee` example, the *payroll feature* would typically rely on attributes such as salary scale and number of hours worked, and on functions that access those attributes to compute the corresponding salary/wages. Similarly, a *production planning* feature would rely on attributes such as skill sets, work schedule, and number of hours worked, and on functions that access such attributes to schedule an employee to work on a particular shift, doing a particular task.

Our overall study is aimed at developing methods for identifying sets of classes, attributes and methods that might support the implementation of a distinct functional feature. In one approach [3], code slicing was used to extract the classes, data members, and methods– actually, statements in the method bodies– that contributed to the return value of a particular function (e.g., computing the wage of an employee). In the line of work presented here, we ignore code-level relationships that may be output by a dynamic analysis of the application source code (e.g. as represented in call graphs, references, etc.) and limit our focus to

the sole information that may be extracted from class signatures (methods and attributes) as an input for the identification of functional features.

Before talking identification, we have to examine the way multiple functional features manifest within the same code base, i.e. consider how a developer might handle several features and interweave them into a single class hierarchy. The separate development and/or packaging of functional features within the same classes/class hierarchies has been extensively studied for the past two decades. In fact, techniques built on top of the OO paradigm for handling of the so called separation of concerns have proliferated (see e.g. [4,5,6,7]). They have come to be collectively referred to as *aspect-oriented development techniques*, named after the Kiczales et al's *aspect-oriented programming* [8]. However, in the absence of these techniques, developers have to resort to design and coding patterns to implement several functional features within the same class hierarchy.

We identified three categories of such implementations. The first one employs *multiple inheritance*: Each functional feature is represented by its own class hierarchy and a class combining several features simply inherits them from the corresponding classes. In the second pattern, features are interweaved by means of *aggregation*: A separate class (hierarchy) represents each functional feature whereas a multi-feature class has an aggregated class for each possessed feature whom it delegates the feature-related services. The third pattern, hereafter called *ad-hoc implementation*, refers to situations where no care was taken to separate the functional features, i.e. no specific structural regularity can be observed involving class relationships like the specialization/aggregation in the previous cases (see an example in Fig. 1.c). In the corresponding classes, the state and behavior of multiple functional features are directly "inlined" in the class with no structural boundaries between them. Nevertheless, even in such structurally unconstrained implementations, some regularities may still be observed, when nothing else, at least in breaking the design rules of thumb.

Fig. 1 shows some Java patterns for implementing multiple functional features. In the intended application, we want to implement the concept of a `PartTimeStudent`, which combines the features (behavior) of a `Student` and a `Worker`. In the first case (Fig. 1.a) the class `PartTimeStudent` implements the interfaces `Student` and `Worker`, i.e. it gets no implementation for the comprised methods. In Fig. 1.b, `PartTimeStudent` inherits from the *class* `Student` and implements the *interface* `Worker`. In the third case (Fig. 1.c), the class `PartTimeStudent` supports the functionalities of `Student` and `Worker` through a combination of implementation inheritance, and delegation.

Naturally, when studying a legacy application, it is not known beforehand whether it embodies different functional features, and if it does, which of the above implementation approaches has been used. Therefore, both the presence/absence of functional features, and the composition technique(s) used to intertwine them, are target hypotheses for our methods. In a separate branch of this study, we proposed techniques for identifying functional features implemented with multiple inheritance and aggregation [9] as depicted in Fig. 1. In this paper, we focus on ad-hoc implementations of multiple functional features.

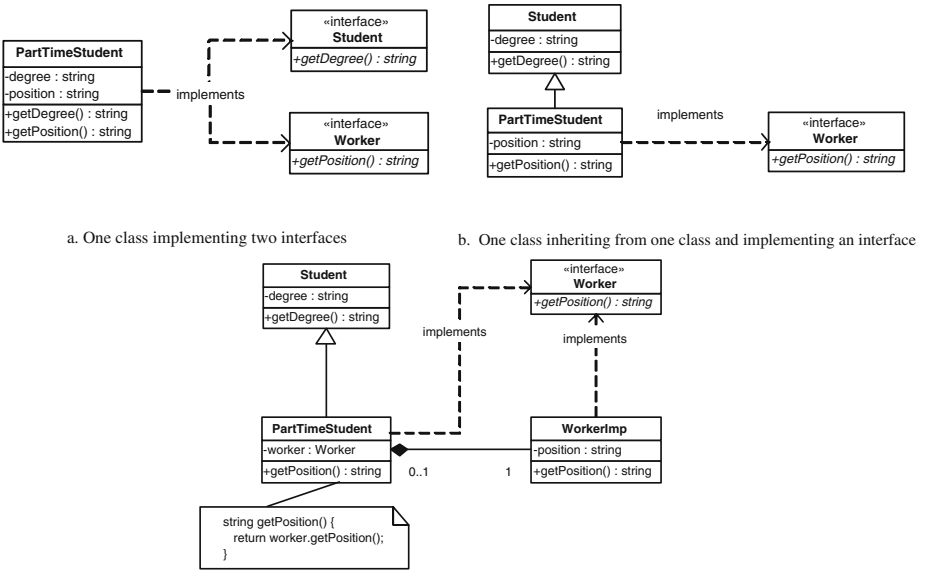


Fig. 1. Some Java patterns for implementing multiple functional features

3 Ad-Hoc Implementations of Multiple Functional Features

By *ad-hoc implementation* we refer to cases where the developer took no special measure to separate the artifacts implementing a particular functional feature from the rest of the class hierarchy. In this case, class members that implement the feature– e.g., related to skills, shifts, number of hours worked, for production planning– are simply in-lined in the classes supporting the feature. How, then, do we recognize that a given set of class members (data and function) contribute to a functional feature? In previous work of the third author, with promising results [3], def-use graphs and code slicing were used to identify the set of classes and members that contribute to computing a single quantity (e.g. an employee's salary). Short of performing a fine-grained code analysis, we rely here on signature-level manifestations of features occurrences.

The basic premise of our approach is that a functional feature might be represented either *intensionally* or *extensionally*. In the first case, the members contributing to a functional feature will be factored into dedicated classes or interfaces (as shown in Fig. 1). These being available throughout the application, the corresponding feature can then be taken advantage of by inheritance or delegation wherever necessary. In contrast, the extensionally represented functional features occur in an expanded form: The contributing members must be defined

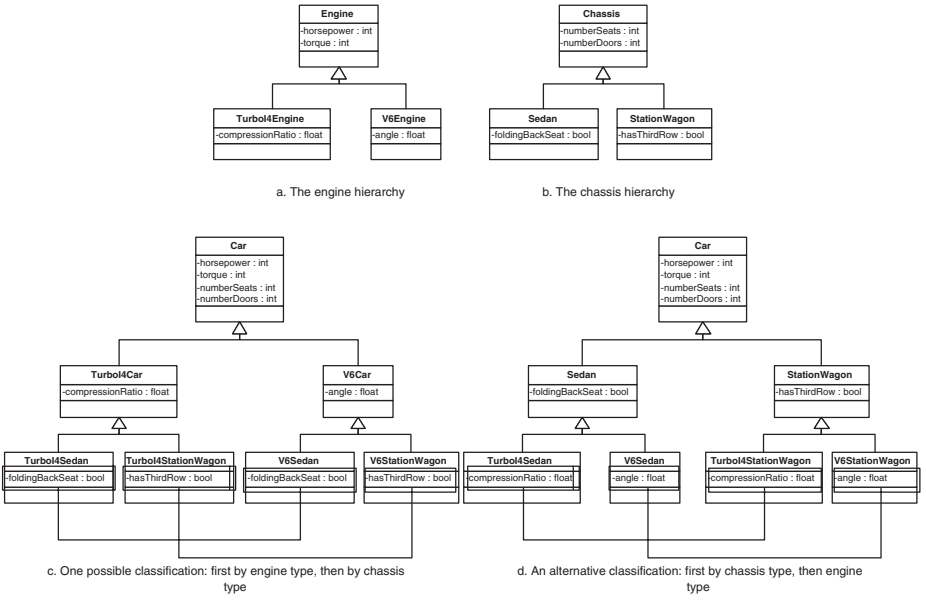


Fig. 2. Supporting multiple functional aspects with state multiplication

at every place of the class hierarchy/tree– or class forest– where the feature is applied. Consequently, the key to recognizing that a set of function/data members embodies a functional feature is to have that set occur in *several places*.

Consider first a special case of ad-hoc implementation –that we call *state multiplication*– to understand the intuition behind this concept before moving to a more general characterization thereof. *State multiplication* is best understood in the context of a situation where several features are to be combined, each coming in different flavors. For instance, cars come in many body types, including sedans, coupe, station wagon, etc. and have various power plants. Figs. 2.a and 2.b show what the respective hierarchies might look like. A given car will thus have a combination of both features. Figs. 2.c and 2.d show two possible car classifications. To clarify our use of "state multiplication", consider the class `Car`, which in both Figs. 2.c and 2.d includes the sum of the "state variables" of the chassis and engine components (all combinations are present).

Yet is such a design plausible? Most likely, if a developer is given the `Chassis` and `Engine` type hierarchies *beforehand*, she/he will try to combine them, by using either aggregation or multiple inheritance. In contrast, hierarchies such as the ones in Figs. 2.c and 2.d could well result from a step-wise design, i.e. starting with the first feature of the problem space, and then specializing the leaf nodes of the class hierarchy based on the second feature.

In [9], the cases of state multiplication were shown to have precise characterizations, e.g. *cartesian* or *tensor product* of feature hierarchies, that lend themselves to efficient factorization algorithms (e.g. see [10]). Yet in real-world applications, such exhaustive –hence easy to factorize– feature combinations are unlikely to

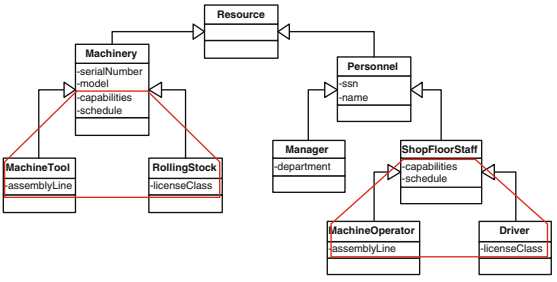


Fig. 3. General case of ad-hoc implementation

occur. Indeed, perfect symmetry may be broken due to a core functionality embodied by the supporting class hierarchy. Moreover, some combinations might be impossible, impractical, or simply prove to be economical no-starters, e.g. no demand for a station wagon with a turbo-I4 engine.

Instead, we focus on a more general case as depicted in Fig 3. A functional feature is recognized by the occurrence of a *structural pattern* of data and function members in different places in the hierarchy, that –we guess– impart a particular behavior on the classes to which they attach. In this case, the functional feature represents what it means to be a *production* resource: Such a resource has `capabilities` and a `schedule`; it specializes into assembly line resources (machine tools and operators), and transportation ones (rolling stock and drivers).

In summary, we hypothesize that a pattern of data and function members that occurs in many places of a legacy application code will, most likely, reflect a functional feature, which, albeit supported by multiple occurrences, would have not been identified as such by the developer. Recurring patterns are typically mined out by means of factorization techniques such as the ones based on formal concepts analysis (FCA) [11] the we explore below. FCA makes emerge conceptual abstractions out of a collection of individuals with properties hence it perfectly fits the tasks of analysis/refactoring of OO applications [12,13].

4 Detecting Functional Feature Occurrences

We recall the typical translation of a class hierarchy in FCA terms and list the differences with the functional feature detection problem before presenting an appropriate encoding thereof and the ensuing mining method.

4.1 Concepts Analysis of OO Class Hierarchies

(Formal) concepts analysis (FCA) [11] addresses the construction of conceptual abstractions, or *concepts*, out of a collection of individuals described in terms of properties. The concepts, which are basically intentionally described clusters, emphasize commonalities in the descriptions of participating individuals. Hence FCA is particularly suitable for the discovery of cohesive groups of entities as the ones sought in class hierarchy analysis [13].

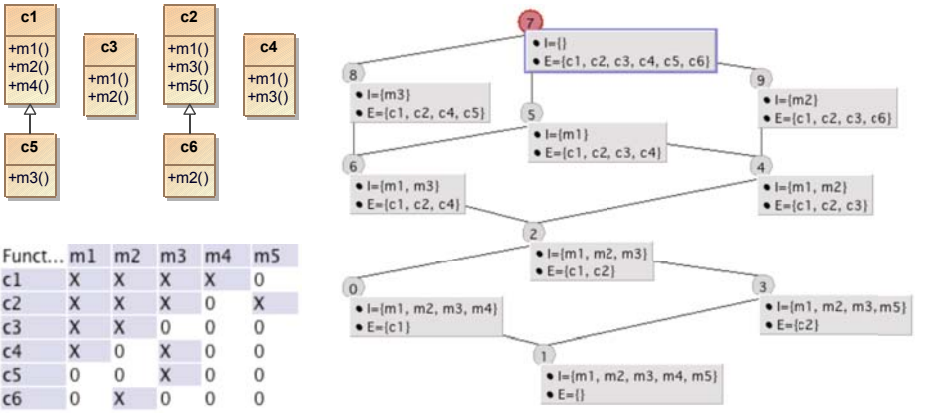


Fig. 4. Left: Initial classes (top) and encoding as context (bottom); **Right:** The concept lattice of the context

In FCA, concepts emerge from a (formal) context $\mathcal{K} = (E, P, I)$ where E is the entity set (formal objects), P the property set (formal attributes) and I (the incidence relation) associates E to P: $(e, p) \in I$ when entity e owns property p. Fig. 4 (in the bottom-left part) provides an example of a context[1] where entities are classes and properties are class function members as drawn in the class diagram in the same Fig. 4 (in the top-left part). Noteworthily, instead of the standard *has-member* relationship, we used for I an alternative encoding of the class hierarchy that we present below. The *concept lattice* of the context, $\mathcal{L}$, is also drawn in Fig. 4 (on the right). For instance, assuming OO refactoring goals [13], a straightforward interpretation of concept 5 in Fig. 4 could be to suggest the design of a new class/interface to host the declaration of $m1()$.

Yet in the feature mining problem as discussed above, the mapping between the class hierarchy components and the elements of a FCA context is all but immediate while the interpretation of the potential concepts is even less so. Indeed, while for refactoring it is crucial to spot common specifications of a set of classes, which puts the emphasis on the members of an individual class, in the present settings, an occurrence of a target feature may be scattered across several classes. Hence the formal objects from the context must be associated to sets of classes rather than to individual classes (with the ensuing increase in the size of the search space for potential feature occurrences).

In a different vein, one might think of exploiting the graph structure of the class hierarchy: the specialization/aggregation links connect classes into a graph while class members provide labels. Furthermore, functional features would correspond to subgraphs of identical structure (i.e. isomorphic) and partially overlapping labels. In such settings, the identification of functional features could be approached as a repeating subgraphs mining problem. Yet, although tempting, this approach is inherently limited as the isomorphism between occurrences of

[1] Empty cells being denoted –unconventionally– by 0.

a functional feature is an overly strong hypothesis. Indeed, such a symmetry cannot be reasonably assumed, not least out of the very same reasons that forbid the perfect state multiplication case. Thus, the only way out seems to lay in a dedicated encoding of the hierarchy to enable a reliable detection of all functional features occurrences (i.e. no false negatives) at a reasonable cost.

4.2 Encoding Class Hierarchies for Functional Feature Mining

Since the occurrences we look for manifest themselves as sets of data/function members, the attribute dimension of our intended translation to FCA is fixed beforehand. This means the functional features will appear in concept intents, as sub-sets thereof. As to the context entities, the natural candidates for that role are subsets of classes from the hierarchy. Yet due to the exponential size of the resulting search space (in the number of all classes), a more compact occurrence representation must be found.

There are various manners to restrict the family of class subsets to effectively examine by means of an FCA-based mining method. For instance, one may require that all classes in a candidate occurrence belong to the same branch of the hierarchy (thus excluding sets like {c1, c2, c4} or {c3, c6} of the above diagram). Conversely there are several ways of associating class members to a set of classes. For instance, in OO refactoring applications of FCA, classes are mapped to formal objects, class members to formal attributes, and incidence to the traditional has-member relationship.

As a good trade-off between precision and search space size, we propose to identify any occurrence with the minimal common super-class(es) of all the classes in the occurrence. This amounts to processing only subsets of classes that represent complete sub-hierarchies of the initial hierarchy, or, technically speaking, *downsets* of the specialization order. Moreover, we only consider downsets that are rooted in a unique class, i.e. subsets made of a root class and all its subclasses. As a result, each formal object can be assimilated to a class from the hierarchy (but represents all its subclasses as well).

In turn, the objects-to-attributes incidence in the context goes to the opposite of the inheritance mechanism: A class is related to its own members as well as to the members of all its sub-classes. The class diagram and the context in Fig. 4 (on the left) show an example of the resulting encoding.

To formalize the above constructs, assume a set of classes C with a universal class $\top$, a hierarchy $\mathcal{H} = \langle C, \leq \rangle$, a set of members M and a membership relation $I \subseteq C \times M$ (cIm, or $(c, m) \in I$ iff the class c defines the method m). Here, we assume identity on methods of identical signatures.

A *functional feature* is a subset of members $N \subseteq M$ whereas a (valid) occurrence thereof is a pair $[A, N]$ where a set of classes $A \subseteq M$ is such that:

- the members incident to A cover N: $N \subseteq \{m | \exists c \in A, cIm\}$, and
- no class is redundant in A since it exclusively contributes at least one member from N: $\forall c \in A, (\exists m \in N : \forall \bar{c} \in A, \bar{c} \neq c \Rightarrow (\bar{c}, m) \notin I)$.

For instance, $(\{c1, c5\}, \{m2, m3\})$ and $(\{c1, c4\}, \{m2, m3\})$ are valid occurrences of $\{m2, m3\}$ whereas $(\{c1, c3, c5\}, \{m2, m3\})$ is not. Obviously, we can only hope

to detect features with at least two occurrences. Please observe that the above property is only a necessary condition as many sub-sets of M that satisfy it clearly do not qualify as functional features.

To restrict the set of candidate occurrences, we shall further require that instead of being scattered throughout the entire hierarchy occurrences represent entire sub-hierarchies. First, an occurrence will be canonically represented by its *roots*, i.e. the minimal common predecessors of its classes: $roots([A,N]) = \min(\{c | \forall \bar{c} \in A, \bar{c} \leq c\})$. To simplify our constructs, we assume a *single-inheritance* hierarchy $\mathcal{H}$, which means $roots([A,N])$ are invariably singletons. For instance, $roots([\{c1,c5\},\{m2,m3\}]) = \{c1\}$ and $roots([\{c1,c4\},\{m2,m3\}]) = \{\top\}$ ($\top$ is the universal class, `Object` in Java). Moreover, we assume *non-trivial* occurrences $[A,N]$ to have roots different from $\top$, whereas two occurrences $[A_1,N_1]$ and $[A_2,N_2]$ are *independent* if $roots([A_1,N_1]) \neq roots([A_2,N_2])$.

As indicated above, we focus on *complete* occurrences $[A,N]$, i.e. where A represents a sub-hierarchy of $\mathcal{H}$ while N gathers the member sets of all classes from A. In such cases, the occurrence comprises its root, $roots([A,N]) \subseteq A$. Yet the roots of an arbitrary occurrence need not to belong to its class set.

To formalize the underlying link, we define a component relation on top of $\mathcal{H}$, M and I. The new relation works dually to the inheritance mechanism hence its name: the *anti-inheritance* of $\mathcal{H}$, M and I is defined as follows: $J \subseteq C \times M$ as cJm iff (1), cIm or (2), $\exists \bar{c} \leq c$ with $\bar{c}Im$. J is the incidence in the context $\mathcal{K} = (C,M,J)$ and will be used as a derivation operator (i.e. we shall write c^J). In Fig. 4, the incidence relation in the context (bottom) is exactly the anti-inheritance of the class diagram (top). For instance, $c1$ is incident to $m3()$ since the latter is defined in a sub-class of the former ($c5$).

For instance, in an arbitrary occurrence $[A,N]$, $c \in roots([A,N])$ implies $N \subseteq c^J$, whereas in a complete one $N = c^J$. Obviously, the context $\mathcal{K}$ comprises all objects corresponding to complete occurrences from the hierarchy.

Now the potential functional features targeted here correspond to groups of at least two independent –yet possibly non complete– occurrences of the same member set N. Due to the specific nature of concept intents, i.e. intersections of the initial class J-images, only maximal N can be identified (hence the manual removal of co-occurring but unrelated members from N may prove necessary).

Moreover, the potential features N are characterized by a distinct structural pattern within the lattice of $\mathcal{K}$, $\mathcal{L}$, around the underlying concept (X,Y) ($Y = N$). On the one hand, the roots of all the occurrences of N within $\mathcal{H}$ belong to X and, what is more, are among its minima, $\min(X)$. As we require at least two independent occurrences, this forces $|\min(X)| \geq 2$. On the other hand, we want to avoid the examination of spurious candidates corresponding to re-occurring parts of the actual features. For instance, there is no point in considering the occurrences of `assemblyLine` corresponding to the classes `MachineTool` and `MachineOperator` in Fig. 3 as both of them are simply parts of the occurrences of the same larger feature.

To that end, we require that interesting concepts are not part of a chain in the lattice following the set inclusion between recursive parts of the same actual

occurrence (due to the monotony of $\_^J$ with respect to $\leq$). In mathematical terms, this amounts to concept (X, Y) not possessing a direct predecessor, (X_1, Y_1), with the same number of minimal concepts in its extent ($|\min(X)| = |\min(X_1)|$).

In our example, the concepts 2, 4, 5, and 6 satisfy the above conditions. For instance, 5 has four classes in its extent, all of them minima, whereas both of its immediate predecessors, 4 and 6, have three minimal concepts. In contrast, 8 fails the test since among the four concepts in its extent, three are minima, the same number as its immediate predecessor 6. The overall interpretation of the example is: There are two functional features, $\{m1, m2\}$ and $\{m1, m3\}$, both having three independent occurrences. Their combination $\{m1, m2, m3\}$ is a compound feature that occurs twice while their intersection, $\{m1\}$ has no independent occurrences, hence it is delicate to decide whether it represents a feature of its own.

4.3 Mining Features Out of the Concept Lattice

Assume now a software piece P with a class hierarchy $\mathcal{H} = \langle C, \leq \rangle$, member set M and incidence relation I. A context $\mathcal{K} = (C, M, J)$ is induced where J is the anti-inheritance as defined above. To extract the target functional features, we analyze the concept lattice of $\mathcal{K}$, $\mathcal{L}$. At a pre-processing step, for each concept $n = (X, Y)$ from $\mathcal{L}$, the minimal classes in its extent, $\min(X)$, are calculated.

The mining algorithm as shown below takes the concept lattice $\mathcal{L}$ as input and outputs the list of candidates (*CandFeatureList* hereafter). For each concept (X, Y), from the lattice top ($T_{\mathcal{L}}$) downwards, the algorithm checks the interestingness conditions, i.e. at least two classes in $\min(X)$ and no immediate predecessor concept with the same number of minimal classes. Concepts satisfying both conditions are kept in *CandFeatureList* for further examination.

```
Input: concept lattice L
Output: feature candidates CandFeatureList
ListConcept ← children(T_L)
CandFeatureList ← ∅
while ListConcept ≠ ∅ do
    (X, Y) ← extract(ListConcept)
    if |min(X)| > 1 then
        add((X, Y), CandFeatureList)
        foreach (X_, Y_) ∈ children((X, Y)) do
            add((X_, Y_), ListConcept)
            if (|min(X_)| = |min(X)|) then
                remove((X, Y), CandFeatureList)
```

5 Preliminary Experiments

To validate our approach, we implemented the algorithm from section 4.3 and carried out an experimental study whose setup and outcome are presented below.

Table 1. "Vital" data about the tested software packages

Software	# LOC	# Classes/Interf.	# Methods
FreeMind	65490	712	4785
JavaWebMail	10707	95	1079
JHotDraw	9419	171	1229
JReversePro	9656	87	663
Lucene	15480	196	1270

5.1 Experimental Settings

We applied our mining tool to a number of open-source Java applications. The selected software covers a wide range of application areas: a graphical editing framework (JHotDraw), a tool for reverse engineering compiled Java code (JReversePro), a web mail client (JavaWebMail), an information retrieval framework (Lucene), and a mind mapping tool (FreeMind). The applications have different maturity levels, going from version 0.7.1 for FreeMind, to version 5.1, for JHotDraw. This, combined with the type of the application– system-type applications versus domain applications– meant that the applications did not have the same design quality. Table 1 provides some quantitative data about these systems: lines of code, number of classes/interfaces and of methods.

We used Eclipse's JDT API to analyze the legacy code in order to extract class signatures (the set of public data members and public functions) and class relationships. The latter comprise extension relationships between classes, between interfaces, `implements` relationships between classes and interfaces, and aggregation relationships between classes (i.e. cases where class A has a data member whose type is a class B). We excluded from our analysis the code that comes from common libraries. For example, when analyzing the JHotDraw application the `JComponent` class was ignored, even though some JHotDraw classes inherit from it. As we shall see, such cases can be filtered out easily.

5.2 Experimental Results

Table 2 shows the size of the lattice as discussed in section 4.2 (in the 2nd column labeled # conc.), the total number of candidate features (# cand.), i.e. concepts satisfying the interestingness criteria, and the number of candidate features with *at least* two methods. Table 2 also quantifies the outcome of a manual examination of the candidate features which were found to fall into four categories (presented in details below): features inherited from an interface/superclass (I/C) or acquired by delegation (D) jointly account for known ones (# known feat.). Unknown features (# unknown feat.) are split into ad-hoc (ad-hoc) or "different structures" implementation (DS).

Implementation per inheritance covers candidate features involving an implemented interface or a superclass. These satisfy the properties in section 4, yet do not represent a recognizable functional feature, or one worth packaging.

Table 2. Output metrics for the experimental set of software systems

Software	# conc.	# cand.	# >= 2 meth.	# known feat.		# unknown feat.	
				I/C	D	ad-hoc	DS
FreeMind	1403	167	103	15	8	60	14
JavaWebMail	713	49	36	10	0	26	0
JHotDraw	581	174	147	21	2	13	1
JReversePro	116	19	10	1	0	9	0
Lucene	98	66	47	14	3	15	4

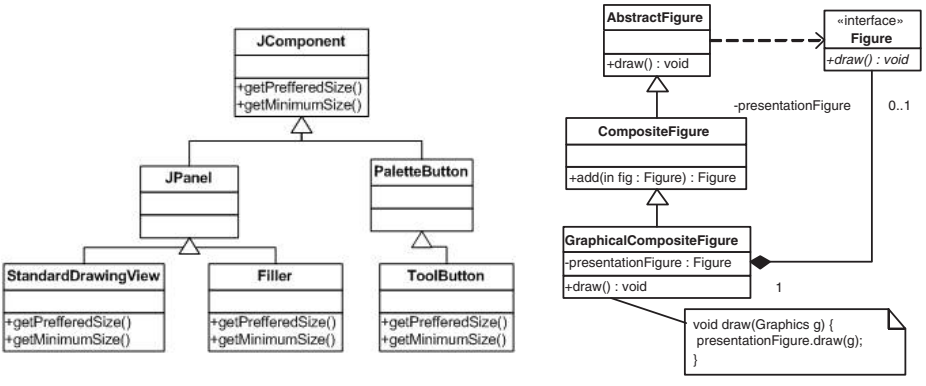


Fig. 5. JHotDraw: Inheritance-based implementation (left) and delegation (right)

An interesting case is one where we "inadvertently" misled the algorithm by forcing classes from external libraries to be discarded. As a result, in a number of cases, the algorithm signaled several independent occurrences of a "feature", whereas, as it turned out after a manual examination, the underlying set of members were actually inherited from a common external superclass that was ignored. For instance, in JHotDraw, the methods `getPreferredSize(), getMinimumSize()` appear in the classes `ToolButton`, `StandardDrawingView`, and `Filler` (see diagram on the left in Fig. 5) which have no common superclass in the local class hierarchy. However, if one follows their respective specialization links beyond JHotDraw, one reaches the Java Swing `JComponent` class. Noteworthily, such cases are easily eliminated by a post-process scanning of candidate classes for common ancestors.

Deliberate implementations of multiple features These are candidates that did correspond to actual functional features, yet ones that developers had deliberately captured as such, and that our algorithm caught. Indeed, the algorithm was designed to skip the most popular coding patterns used to design and package functional features (see Fig.1), in particular, cases of features packaged as interfaces. For example, JHotDraw has the interface `DrawingView`, which is implemented by `StandardDrawingView` and `NullDrawingView`, and thus, *a-fortiori*, `StandardDrawingView` and `NullDrawingView` will implement a substantial set of common methods. Yet our algorithm ignored this case, as it should. However,

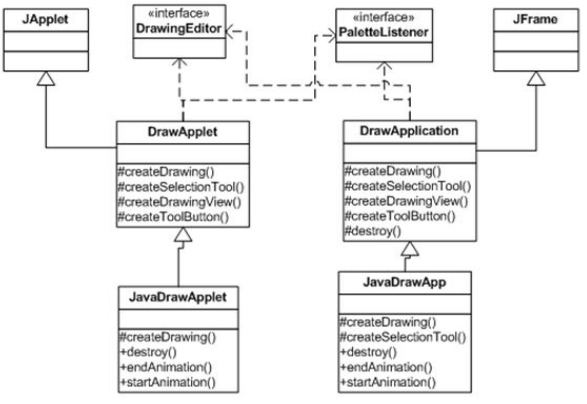


Fig. 6. A case of ad-hoc multiple feature implementation in JHotDraw

one case escaped us– and the algorithm– and is illustrated in Fig. 5, on the right. In fact, while our algorithm takes care of inheritance (if A inherits from B, A and B cannot be counted as independent occurrences), and the `implements` relationship (if A and B implement the interface C, A and B cannot be counted as independent occurrences), it misses subtleties about their combination, e.g. that implements relationships are inherited: if A extends B, and B implements C, then A also implements C. Because of this omission, the algorithm presented us with the methods of interface `Figure` as occurring independently in `Figure` and `GraphicalCompositeFigure`.

Admittedly, the case in Fig.5 can be easily fixed. However, we do expect that less mature frameworks than `JHotDraw` might have "imperfect" implementations of delegation where the component and the composite do not, explicitly– directly or indirectly, as in Fig. 5– implement the same interface.

Ad-hoc implementations of functional features These correspond to the focus of our approach. The algorithm identified a number of situations where two or more classes shared a significant subset of their API without there being any relationship between them, either directly (inheritance) or indirectly, via a common implemented interface. This is a case where a developer, who probably recognized the behavioral similarity, was disciplined enough to choose the same method signatures in both places, but stopped short of formalizing that similarity by formalizing it into a common interface, or a common ancestor.

Fig. 6 shows an example. In JHotDraw, the `DrawApplet` class is to execute the application in a browser (Applet), `DrawApplication` in a window (Frame). The methods in classes `DrawApplet` and `DrawApplication` are exactly *identical*, the same for the sub classes `JavaDrawApplet` and `JavaDrawApp`. Here the developers could have gathered the shared methods in a common superclass of `DrawApplet` and `DrawApplication`, named, say, *DrawCreation* in order to achieve better factorization. A sort of extreme example is shown in Fig. 7: the depicted

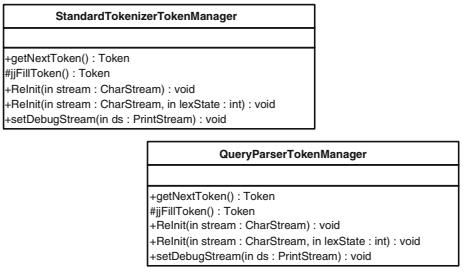


Fig. 7. Another case of ad-hoc multiple feature implementation in Lucene 1.4

classes have *identical* APIs (there are no methods except the ones shown). Thus, both classes embody *what it means to be a token manager*, applied to the parsers of the Lucene query and of the the input text, respectively.

Different structures of functional features The feature can be recognized by the occurrence of a structural pattern of function members in different places in a hierarchy such as the one in Fig. 3, or the feature is the set of methods scattered across a set of classes. These may take a variety of forms. For space limitation reasons, we skip the presentation of this category.

6 Related Work

The recent work on feature location (after 2000) can be roughly divided into three broad categories (see e.g. [14]): 1) methods based on lexical analysis of the source code (e.g. the *Aspect Mining Tool* [15], and to some extent, FEAT [16] and [2,17,18]); 2) methods based on a static analysis of the code, using structures such program dependency graphs, call graphs, and the like (e.g. [19], [20]); and 3) methods based on a dynamic analysis of the code, using things such as event or execution traces (e.g. [21,22,23]); this is a summary characterization as most approaches involve some mix of techniques (e.g. lexical and static/type analysis, or static and dynamic). Our approach falls in the static category, because we perform feature identification based on class signatures. However, our method is much simpler to implement as it does not involve complex static analyses.

In terms of focus, most of the existing work focuses on aspect-like features which tend to correspond to infrastructure or architectural services. This influences both the granularity of the analysis– method code– and the type of patterns that we look for– for example, performing a fan-in analysis [19] or looking for code clones [20]. Our definition of *functional features* corresponds best to features in [21], which embody externally visible behavior of the objects.

Finally, FCA has been used by a number of methods (e.g. [21,23,18]) as it provides an elegant formalism for identifying cohesive sets of items in the context of a degenerate (many-to-many) relationships. Our use of FCA is perhaps closest to that in [21], which clusters *code units* based on their participation in features. A major difference, however, between our method and theirs is that, Eisenbarth

et al rely on the input of a human expert to, a) specify the features we are looking for, and b) design execution scenarios that exercise those features. In our case, the identification of the feature is, in and of itself, an output of our algorithm. The difference between the two methods is similar to that between supervised learning– the case of [21]– and unsupervised learning– our case.

7 Conclusion and Future Work

AOSD improves upon OO development by providing programming abstractions that enable us to separate a wider range of functional requirements into their own development artifacts. Our work deals with the identification of such functional features in legacy OO applications. Minimally, this identification should make it easier to understand and maintain the application. Ideally, this could be the first step toward aspect refactoring.

In our work on mining functional features from OO code we explore various hypotheses about how developers, absent AOSD abstractions, would implement several functional features within the same code base. In this paper, we studied the case of an *ad-hoc* implementation of multiple functional features, which corresponds to situations where the developer did not realize the presence of these different features, or simply, did not know any better. Our approach for recognizing such features consists in identifying groups of class members (methods and attributes) that occur in separate places in the class hierarchy. The results of the preliminary experiments suggest that our method detects both *ad-hoc* and *deliberate* implementations of features. Generally speaking, the algorithm proved to be a powerful abstraction mechanism for identifying regularities– and irregularities– in the design. Our work continues on refining the algorithm by taking advantage of the structure of the data, and by relaxing our definition of feature to be able to handle "near occurrences".

References

1. Mili, H., Sahraoui, H., Lounis, H., Mcheick, H., ElKharraz, A.: Concerned about separation. In: Baresi, L., Heckel, R. (eds.) FASE 2006. LNCS, vol. 3922, pp. 247–261. Springer, Heidelberg (2006)
2. Marcus, A., Poshyvanyk, D.: The conceptual cohesion of classes. In: Proc. of ICSM 2005, pp. 133–142 (2005)
3. Dagenais, B., Mili, H.: Slicing functional aspects out of legacy code, 10 p. (2008) (submitted)
4. Hailpern, B., Ossher, H.: Extending objects to support multiple interfaces and access control. IEEE Trans. Softw. Eng. 16, 1247–1257 (1990)
5. Aksit, M., Bergmans, L., Vural, S.: An object-oriented language-database integration model: The composition-filters approach. In: Lehrmann Madsen, O. (ed.) ECOOP 1992. LNCS, vol. 615, pp. 372–395. Springer, Heidelberg (1992)
6. Harrison, W., Ossher, H.: Subject-oriented programming (a critique of pure objects). In: Proc. of ACM OOPSLA 1993, vol. 28, pp. 411–428 (1993)

7. Tarr, P., Ossher, H., Harrison, W., Sutton, S.: N degrees of separation: multi-dimensional separation of concerns. In: Proc. of ICSE 1999, pp. 107–119 (1999)
8. Kiczales, G., Irwin, J., Lamping, J., Loingtier, J., Lopes, C., Maeda, C., Mendhekar, A.: Aspect-oriented programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
9. Elkharraz, A., Mili, H., Valtchev, P.: Mining functional aspects from legacy code. In: Proc. of ICTAI 2008, pp. 403–412. IEEE Comp. Soc., Los Alamitos (2008)
10. Aurenhammer, F., Hagauer, J., Imrich, W.: Cartesian graph factorization at logarithmic cost per edge. Computational Complexity 2, 331–349 (1992)
11. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin (1999)
12. Godin, R., Mili, H., Mineau, G., Missaoui, R., Arfi, A., Chau, T.: Design of Class Hierarchies based on Concept (Galois) Lattices. TAPOS 4, 117–134 (1998)
13. Godin, R., Valtchev, P.: Formal concept analysis-based normal forms for class hierarchy design in OO software development. In: FCA: Foundations and Applications, pp. 304–323. Springer, Heidelberg (2005)
14. Revelle, M., Poshyvanyk, D.: An exploratory study on assessing feature location techniques. In: ICPC 2009, Vancouver, BC, Canada (2009)
15. Hannemann, J., Kiczales, G.: Overcoming the prevalent decomposition of legacy code. In: Workshop on Advanced Separation of Concerns, ICSE 2001, Toronto (2001)
16. Robillard, M., Murphy, G.: Concern graphs:finding and describing concerns using structural program dependencies. In: Proc of ICSE 2002, pp. 406–416 (2002)
17. Shepherd, D., Fry, Z.P., Hill, E., Pollock, L., Vijay-Shanker, K.: Using natural language program analysis to locate and understand action-oriented concerns. In: Proc. of AOSD 2007, USA, pp. 212–224. ACM, New York (2007)
18. Poshyvanyk, D., Marcus, A.: Combining formal concept analysis with information retrieval for concept location in source code. In: Proc. of ICPC 2007, pp. 37–48 (2007)
19. Marin, M., van Deursen, A., Moonen, L.: Identifying aspects using fan-in analysis. In: Proc. of WCRE 2004, USA, pp. 132–141. IEEE Computer Society, Los Alamitos (2004)
20. Shepherd, D., Gibson, E., Pollock, L.: Design and evaluation of an automated aspect mining tool. In: Proc. Intl. Conf. on Soft. Eng. Research and Practice (2004)
21. Eisenbarth, T., Koschke, R., Simon, D.: Locating features in source code. IEEE Trans. Software Eng. 29, 210–224 (2003)
22. Breu, S., Krinke, J.: Aspect mining using event traces. In: Proc. of ASE 2004, Washington, DC, USA, pp. 310–315. IEEE Computer Society, Los Alamitos (2004)
23. Tonella, P., Ceccato, M.: Aspect mining through the formal concept analysis of execution traces. In: Proc. of WCRE 2004, pp. 112–121 (2004)

Concept Neighbourhoods in Lexical Databases

Uta Priss and L. John Old

Edinburgh Napier University, School of Computing
j.old@napier.ac.uk
www.upriss.org.uk

Abstract. This paper discusses results from an experimental study of concept neighbourhoods in WordNet and Roget's Thesaurus. The general aim of this research is to determine ways in which neighbourhood lattices can be derived in real time from a lexical database and displayed on the web. In order to be readable the lattices must not be too large, not contain overlapping concepts or labels and must be calculated within seconds. Lattices should, furthermore, not be too small and they should contain sufficient complexity to be interesting for the viewer. For these purposes the sizes of the lattices of different types of concept neighbourhoods have been calculated. Using the size information should help with the task of on-line generation of the lattices.

1 Introduction

Concept neighbourhoods are a means of extracting smaller-sized formal contexts from a larger formal context whose concept lattice is too large to be viewed as a whole. The corresponding *neighbourhood lattices* consist of a concept and its neighbours. Roget's Thesaurus (RT) is an example for which the extraction of concept neighbourhoods has been studied in some detail (Priss & Old, 2004 and 2006). An on-line interface at www.roget.org lets users explore concept neighbourhoods of Roget's Thesaurus in real-time. The algorithm for constructing the neighbourhoods uses a number of heuristics which ensure that the lattices are neither oversized, nor trivial.

The goal of our current research is to implement a similar interface for WordNet (Fellbaum, 1998) and potentially for other lexical databases in the future. Previous research has shown that the formation of concept neighbourhoods and the establishment of heuristics for generating reasonably-sized lattices depend on the structure of the underlying resources. For example, Dyvik's (2004) method for constructing a thesaurus from a bilingual corpus (which is very similar to our method of building concept neighbourhoods) does not work so well if a bilingual dictionary is used instead of a corpus (Priss & Old, 2005). The reason for this is that the translational relations between words in a corpus show more diversity than in a dictionary. Thus, even if a lexical database has a similar hierarchical structure to Roget's Thesaurus, the algorithms for forming concept neighbourhoods may require some adjustment. Furthermore, if a database such as WordNet contains a variety of semantic and lexical relations, it seems reasonable to attempt to incorporate these existing relations into the formation of concept neighbourhoods.

WordNet (Fellbaum, 1998) is a lexical database which groups words into *synsets* of synonyms (or near synonyms). Each synset belongs to a part of speech (noun, verb,

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 283–295, 2010.

adjective, adverb) and can participate in several part-of-speech-dependent semantic and lexical relations. For example, the semantic relations for nouns are IS-A relations (hypernymy/hyponymy) and several types of part-whole relations (meronymy). In contrast to semantic relations which are defined between synsets, lexical relations (such as antonymy) are defined between words. This is because the designers of WordNet took a psychological perspective and decided that antonymy is dependent on word associations. For example, in the synset "large, big", "large" is an antonym of "small" and "big" is an antonym of "little" because people tend to associate these. From a logical perspective it can be argued that antonymy simply expresses a form of contrast or opposition that can be applied to the whole synset. Therefore in our applications we sometimes generalise a lexical relation (between two words from different synsets) into a semantic relation (between all words of the two synsets). Alternatively it is also possible to treat a semantic relation between two synsets as a lexical relation between all words of the two synsets.

WordNet has been used in many research projects and has been visualised in a number of formats. Probably the most well-known visualisation is the one at visualthesaurus.com, which employs Java-applets to draw networks around words using WordNet's semantic relations and the spring-embedder algorithm. This visualisation differs from our research because it does not result in lattices, but in networks, which completely ignore the hierarchical structure of WordNet's relations. WordNet has been used in several Formal Concept Analysis projects for example by Hotho et al. (2003) as a means for improving text clustering by exploiting WordNet's semantic relations and by Martin & Eklund (2005) for adding hypernymic concepts in a lattice of a semantic file system. But in these projects both WordNet and FCA are just tools used for other purposes and, again, this kind of research differs from what we are intending to do.

Also of interest are studies that compare WordNet and Roget's Thesaurus (for example, Old (2003)). Kennedy & Szpakowicz (2008) discover that different editions of Roget's Thesaurus are quite similar to each other and to WordNet with respect to the calculation of semantic similarity and their usage in language-based information retrieval enhancement methods. Therefore it should be expected that concept neighbourhoods extracted from Roget's Thesaurus and WordNet are similar in structure and size.

Section 2 discusses requirements of an on-line interface for concept neighbourhoods. Section 3 introduces the notions of concept neighbourhoods and neighbourhood lattices in more detail. Section 4 presents examples of concept neighbourhoods from WordNet. Section 5 discusses experimental results that were conducted on the WordNet database and in comparison with Roget's Thesaurus.

2 An On-Line Interface for Concept Neighbourhoods

Creating an on-line interface that generates concept neighbourhoods on the fly poses a number of challenges. Both WordNet and Roget's Thesaurus (RT) contain more than 100,000 words. It would therefore be inefficient to generate all neighbourhood lattices in advance and store them as image files. Furthermore, an interface should allow for a certain amount of flexibility. It should offer choices of semantic relations to be included, degrees of size restrictions, and so on, because there are many possibilities for creating

concept neighbourhoods, and users may have different interests and preferences. The idea is to create an interface similar to Dyvik's (2004) semantic mirrors interface, which allows users to choose different data sources, thresholds and limits.

The technology we are using at the moment consists of MySQL databases of WordNet and RT and the FcaStone[1] software, which uses Graphviz[2] for computing the graph layouts. The database for RT was created by the second author (based on work described by Sedelow & Sedelow (1993)). The WordNet database is built using WordNet SQL Builder[3].

Requirements for generating on-line lattices are that the lattices should:

- be easy to read, not too large, and not too complex so that they are viewable without major zooming or scrolling;
- contain no overlapping nodes or labels;
- be generated within seconds.

Addressing the last requirement first: FcaStone's performance declines sharply when one tries to compute lattices with more than 100 concepts, but smaller lattices can be computed within seconds. Because lattices with more than 50 concepts are not very readable anyway, the main limit for the size of the lattices is readability, not software performance.

Nodes and labels should not overlap. An automatically generated layout of a lattice needs to determine the placement of the concepts, and also the placement of the labels. Since Graphviz does not provide a separate option for placing labels, we are representing each concept as a box which contains the attributes in the top half and the objects in the bottom half (as in Figure 1). Both attributes and objects are limited to 30 characters. If there are more objects or attributes, the string is truncated at the 30th character and dots (...) are inserted at the end. In that way, it can be guaranteed that neither concepts, nor labels overlap anywhere.

To some degree the readability depends on the structure of the lattice. Because of the placement of the labels inside the concept boxes, only up to 10 concepts (or less in smaller browser windows) can be displayed side by side. Graphviz sometimes draws the edges slightly curved, not straight, and in longer anti-chains and crowns, the edges may touch each other or overlap and become difficult to trace. Thus, lattices with the same number of concepts can have very different readability: a lattice with 20 concepts which are arranged in 4-5 levels may be easier to read than a lattice with 20 concepts which contains an anti-chain with more than 10 concepts.

3 Concept Neighbourhoods and Neighbourhood Lattices

A concept neighbourhood is extracted by starting with one item (word, term or concept) and then retrieving other items the first item is related to and so on. This is called the *plus operator* (Priss & Old, 2004) and is usually applied to a formal context of objects

[1] http://fcastone.sourceforge.net
[2] http://www.graphviz.org
[3] http://wnsqlbuilder.sourceforge.net

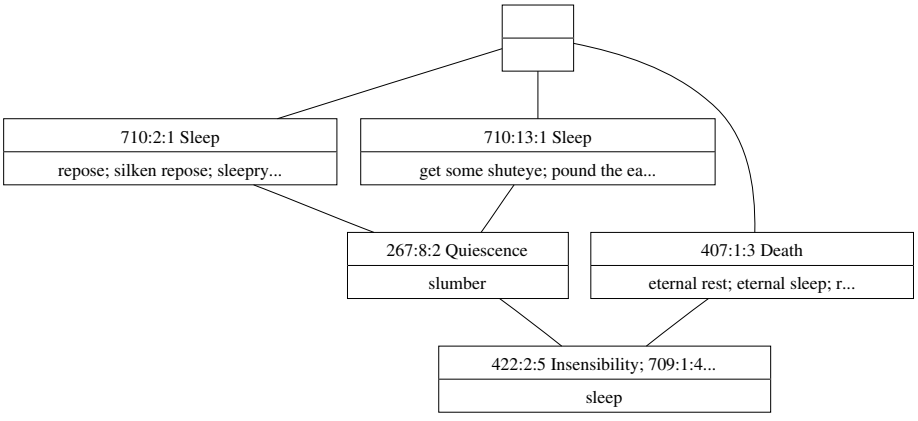


Fig. 1. A neighbourhood lattice in RT for the word "sleep"

and attributes, such as words and their translations into another language, words and their sense numbers in Roget's Thesaurus, or documents and their classification codes. The complete formal context of a lexical database might contain more than 100,000 rows and columns. This is too large to build a readable lattice diagram. Starting with one object or attribute the plus operator is usually applied for a fixed number of times, because if the plus operator is applied an unlimited number of times the neighbourhood might grow until it encompasses the complete or nearly complete lexical database. Apart from stopping the plus operator after a fixed number of steps, it is also possible to apply several restriction methods (Priss & Old, 2004) in order to prevent the concept neighbourhoods from becoming too large.

A *plain n-m-neighbourhood* starts with an object and has the plus operator applied $(2n - 2)$-times to obtain the set of objects and $(2m - 1)$-times to obtain the set of attributes. Thus, a 2-1-neighbourhood of WordNet or RT consists of all the words in the synsets of a word and the senses of the original word. A 2-2-neighbourhood consists of all the words in the synsets of a word and all of their senses. Figure 1 shows a 2-1-neighbourhood lattice for the word "sleep" in RT. The plus operator was applied twice: first, to find all the senses of "sleep" (as formal attributes) and then one more time to find other words (as formal objects) which have the same senses. The senses are described numerically as "Category number:Paragraph number:Synset number" followed by the head word of the category. For example, "710:2:1" and "710:13:1" are two senses, both belonging to the category "Sleep". The layout and design of the lattice was generated automatically as described in the previous section.

4 Concept Neighbourhoods in WordNet

WordNet contains many different types of relations. Therefore there are many possibilities for creating different types of neighbourhood lattices. One possibility is to ignore

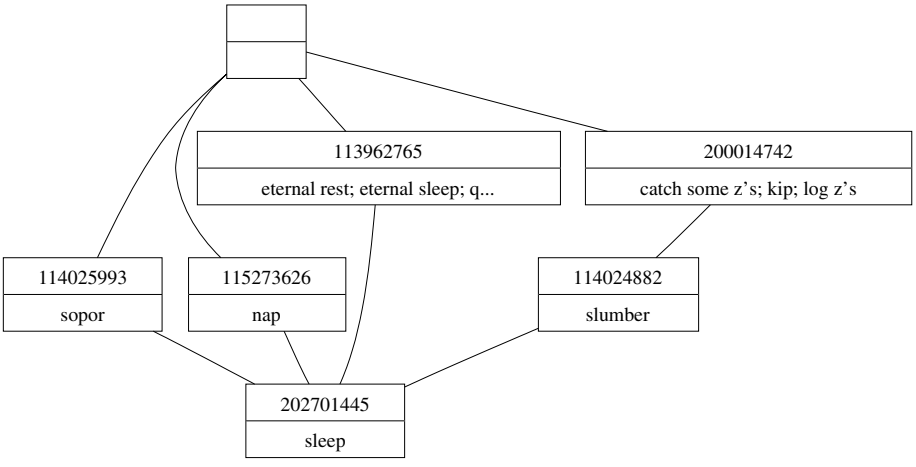


Fig. 2. A lattice of a plain 2-1 neighbourhood in WordNet for the word "sleep"

the semantic relations completely and simply generate plain neighbourhoods exactly as in RT by taking a word, its synsets, all other words in these synsets, their other synsets and so on.

Figure 2 shows the 2-1 neighbourhood for "sleep" in WordNet. The senses are represented by their index numbers. The fact that both RT (in Figure 1) and WordNet contain the words "eternal rest, eternal sleep" in the same sequence in the neighbourhood of "sleep" is not a coincidence because when WordNet was started, the WordNet lexicographers used RT as one of their sources. Thus, WordNet was influenced by RT. Otherwise, the two neighbourhoods only share the words "rest" and "slumber". In other examples we looked at, it also seemed to be the case that only very few words are shared between the WordNet and RT neighbourhoods. We have not yet explored this more systematically, but it might be of interest to calculate the intersection between the WordNet and RT neighbourhoods because they might highlight core synonyms of each word. But that is left for future research.

A 2-1 neighbourhood will only have some interesting structure if the synsets (representing the different senses of a word) have intersections of more than one word. In RT, synsets seem to be fairly large and the synsets corresponding to different senses tend to have larger intersections. In WordNet, however, the synsets tend to be smaller and tend to intersect only in one, sometimes two words. In Figure 2, the only intersection other than "sleep" is the word "slumber" which occurs in two synsets. The reason for this difference between WordNet and RT may be that the lexicographers who create WordNet can view all synsets (senses) of a word when they edit the data. Synsets can be created in a similar manner as traditional dictionaries where the senses of a word are carefully distinguished and balanced. It may be that too much overlap between synsets is deliberately avoided by WordNet's lexicographers. In Roget's Thesaurus, the different senses of a word are only visible in the index, which was created (for the original edition) after the construction of the thesaurus was finished. Roget would not have been able to deliberately control the overlap between synsets. Therefore 2-1-neighbourhoods of

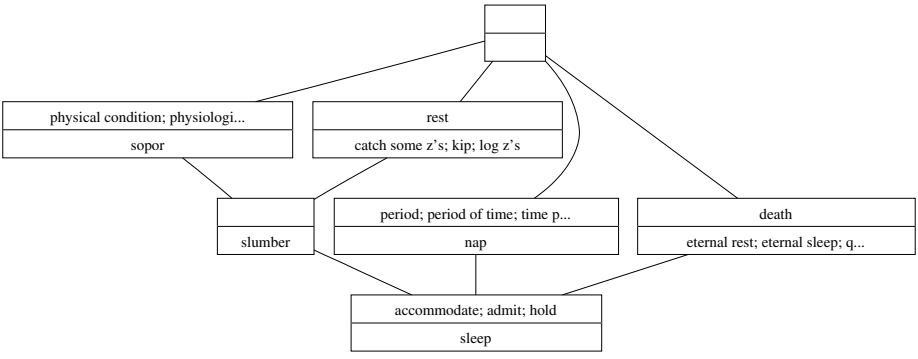


Fig. 3. A lattice of a hypernymy neighbourhood in WordNet

polysemous words (especially nouns) in WordNet are often anti-chains whereas in RT their 2-1-neighbourhoods tend to have more interesting structures. For WordNet other means of creating neighbourhood lattices need to be investigated.

Another possibility is to incorporate WordNet's semantic relations into the building of neighbourhood lattices. One difficulty with this approach is that all parts of speech in WordNet have different types of semantic relations and require different approaches. For nouns and verbs the hypernymy relation can be used as follows: the words of the synsets belonging to all senses of a word are taken as formal objects and the words of the hypernymic synsets are the formal attributes. The relation between objects and attributes is the semantic relation between synsets and their hypernymic synsets but treated as a lexical relation between the words in the synsets. We call this the *hypernymy neighbourhood* because it is based on the hypernymy relation. A *hyponymy neighbourhood* is formed by using hyponymy, and so on. Adjectives and adverbs do not have a hypernymy relation in WordNet, but their "similarity" relation can be used in the same manner.

Figure 3 shows an example of a hypernymy neighbourhood of "sleep" in WordNet. Figure 4 displays the corresponding synsets and their hypernymy relation in WordNet. The example shows that most synsets are maintained as extensions or intensions in a hypernymy neighbourhood. The extensions/intensions are only different from WordNet synsets if the synsets share words (as discussed above for plain neighbourhoods) or share hypernyms. For example, in Figure 3, the left-most concept has the extension "sopor, slumber, sleep" which is not a synset in WordNet. This extension emerges because the two left-most synsets in Figure 4 have the same hypernymic synset. In our opinion, WordNet synsets that share words or hypernyms exhibit implicit structures (in the sense of Sedelow's (1998) "inner structure analysis"), which are detected and visualised when forming hypernymy neighbourhoods. The lattice in Figure 3 is more similar to the RT lattice than the one in Figure 2. But according to our experimental results (see next section), hypernymy neighbourhoods are not in general more similar to RT lattices than plain 2-1 neighbourhood lattices.

Unfortunately, the hypernymy neighbourhoods of most nouns tend to be uninteresting because they tend to be fairly small and predominantly form anti-chains. Hypernymy

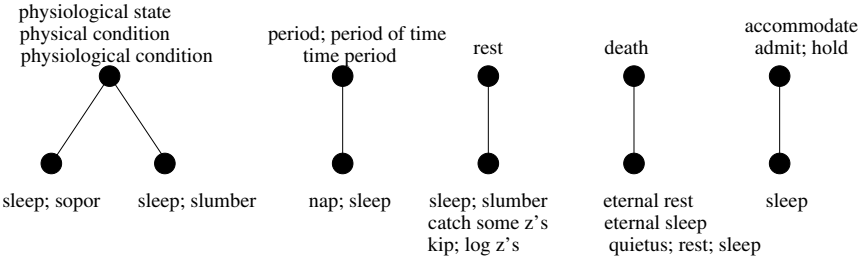


Fig. 4. WordNet's Hypernymy Relation

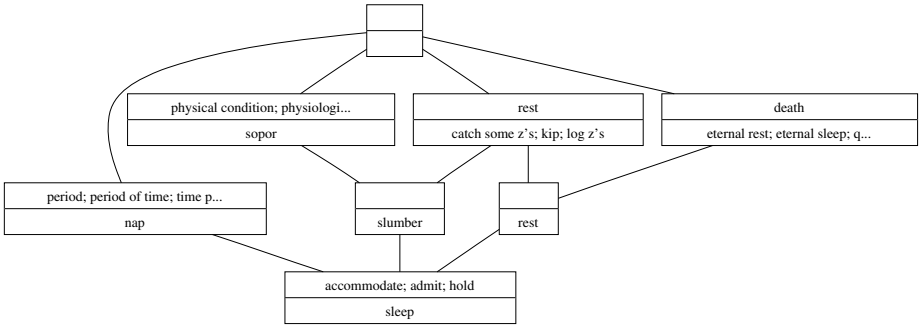


Fig. 5. A Neighbourhood Lattice with Identity in WordNet

neighbourhoods of verbs reveal more structures. The difference between nouns and verbs might be caused either by a structural difference or by the fact that two different lexicographers are responsible for nouns and verbs in WordNet who might use different strategies for implementing hypernymy relations.

Apart from synsets sharing words or hypernyms, it might also be interesting to identify words which occur both as objects and attributes in the same neighbourhood. In Figure 3, the word "rest" occurs both as an attribute and an object (under the attribute "death", but not visible in the Figure because the objects are truncated to 30 characters). An identity relation can be added to the formal context which inserts a cross wherever an object equals an attribute. The result is shown in Figure 5. Another possibility would be to insert crosses whenever there is a substring match between the words in a neighbourhood (matching "rest" and "eternal rest"). These options need to be explored in more detail, but some preliminary analysis indicates that unfortunately, for nouns in WordNet, there is not a significant overlap between objects and attributes.

5 Experimental Results

In order to obtain a better idea as to what kinds of neighbourhood lattices might be most promising for WordNet, we calculated the number of concepts for plain 2-1, plain 2-2 and hypernymy neighbourhoods in WordNet. For comparison we also calculated plain

Table 1. The words with the largest neighbourhood lattices in WordNet and RT and their number of concepts

	WN: hypernymy	WN: plain 2-1	WN: plain 2-2	RT: 2-1	top in both
pass	36	44	582	65	yes
break	39	46	543	44	
take	31	37	657	34	
run	34	39	497	79	yes
hold	31	39	490	47	
set	29	37	336	86	yes
draw	28	30	464	29	
check	27	31	469	52	
make	23	32	549	34	
get	26	28	530	46	
cut	28	36	296	118	yes
go	22	30	561	55	yes
give	26	28	469	17	
place	28	28	321	42	
point	28	29	302	51	
rise	40	42	236	40	
return	28	26	280	40	
turn	23	24	293	87	yes

2-1 and plain 2-2 neighbourhoods in RT. For our test data we chose a list of 45,000 frequently used words. From this list 26,000 words occur in WordNet and 21,000 in RT. The list includes names, placenames and so on, not all of which occur in WordNet and RT. Because the list does not include phrases or compound words, it contains only about 1/4 of the words in WordNet and RT. But phrases and compound words tend to be less polysemous and can be expected to generate smaller concept neighbourhoods. Thus, our test data contains all the interesting words. Because not all nouns and verbs have hypernyms in WordNet and adjectives and adverbs do not have hypernyms at all, only about 17,000 words are used for the hypernymy neighbourhoods.

Table 1 shows the words with the largest neighbourhood lattices in WordNet and RT. The words are sorted by their average ranking in WordNet hypernymy, plain 2-1 and plain 2-2 neighbourhoods. The largest hypernymy and plain 2-1 neighbourhood lattices contain about 40 concepts in WordNet and can be graphically displayed. The largest plain 2-2 neighbourhood lattice in WordNet (for the word "take") contains 657 concepts. In the three types of concept neighbourhoods in WordNet that we looked at, the words tend to be ranked in similar positions.

Table 1 also shows the sizes of the RT plain 2-1 neighbourhood lattices of these words. There is slightly less agreement between the WordNet and RT rankings. The last column "top in both" indicates which words have the largest neighbourhood lattices both in WordNet and RT. The word "pass" is among the 5 largest lattices in all types of neighbourhoods. On the other hand, "turn" which has the second largest neighbourhood lattice in RT has much smaller neighbourhood lattices in WordNet.

Table 2. The adjectives with the largest neighbourhood lattices in RT

	RT: plain 2-1	RT: plain 2-2	rank
vile	48	227	1
fixed	43	201	0.89
soft	43	147	0.77
hard	32	179	0.73
easy	40	126	0.69
proper	32	152	0.67
sad	23	172	0.62

Table 3. The adjectives with the largest neighbourhood lattices in WordNet

	WN: plain 2-1	WN: plain 2-2	rank
hard	14	81	0.83
grim	9	122	0.82
easy	10	106	0.79
fresh	12	86	0.78
big	13	74	0.77
tight	11	86	0.75
soft	14	55	0.73
strong	9	78	0.64
awful	11	56	0.62
just	10	61	0.61
substantial	8	55	0.51

A comparison with Old (2003, p. 183) shows that the words with the largest neighbourhood lattices also tend to be the most polysemous words, i.e. the ones with the most senses. This is of course not surprising because the senses are used for the construction of neighbourhoods. Although individual words can slightly differ with respect to the sizes of their neighbourhood lattices in WordNet and Roget, all of the words with large neighbourhood lattices have some verb senses and tend to be of Anglo-Saxon origin. This is true not just for the words in Table 1, but in general. Words which have only noun or adjective senses have much smaller neighbourhood lattices both in RT and WordNet.

Tables 2 and 3 show a listing just for adjectives. The rank in the tables is calculated as the average of (size of lattice)÷(max size of lattice in this neighbourhood type). It is interesting to observe that many negative adjectives have large neighbourhood lattices. The largest one in RT is “vile”; in WordNet “hard” and “grim”. Among the top ten largest 2-2 neighbourhood lattices for adjectives in RT are “abominable”, “obnoxious”, “odious”, “contemptible” and “despicable”. For some reason their lattices are slightly smaller in the 2-1 neighbourhoods and therefore not in Table 2. The adjectives that have large lattices across all neighbourhood types tend to be short words of Anglo-Saxon origin.

In addition to looking at the words with the largest neighbourhood lattices, we also looked at the size distributions. Figure 6 shows the number of lattices with up to 25 concepts for the three types of neighbourhoods in WordNet and the plain 2-1 neighbourhoods in RT. The number of lattices of size 1 is not a good indication of anything

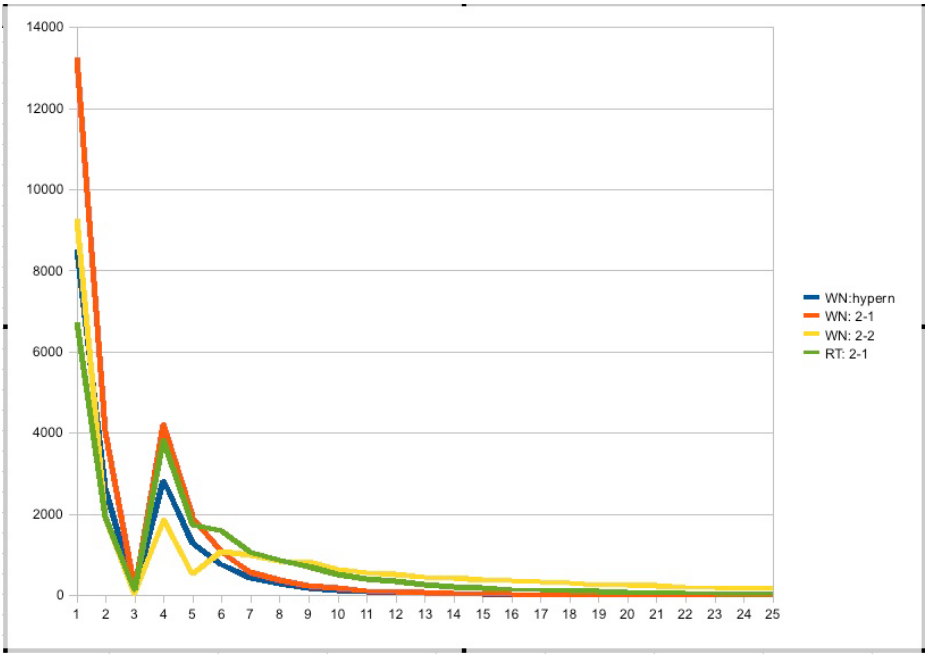


Fig. 6. Number of lattices with up to 25 concepts

because as mentioned before the data sets used for the different tests have different sizes. The set used for RT is smaller than the one used for WordNet. The hypernymy set is smallest because not all words in WordNet have hypernyms. Many of the words with lattice size 1 are proper nouns, abbreviations and other specialised terms. It is to be expected that names and proper nouns have small lattices. There are however some surprises. For example the plain 2-2 neighbourhood lattice of "Adam" in WordNet contains 40 concepts. The reason for this is that "Adam" is a synonym of "ecstasy" and "go" as a hyponym of the drug "mdma". Because "go" is very polysemous, the 2-2 neighbourhood for "Adam" is large too. But such kind of effects are anomalies and indicate homographic or metaphoric word use.

Figure 6 shows that lattices are largest using the plain 2-2 neighbourhoods (which is not surprising because the plus operator is used one more time). The plain 2-1 lattices in RT are larger than in WordNet. The hypernymy neighbourhood lattices are smallest. Figure 6 indicates that the size of the lattices has an impact on the distribution. Normally one might expect to see some kind of power law distribution, which is common for linguistic phenomena (and has been shown to apply to neighbourhood closure lattices in RT by Priss & Old (2006)). But lattices of size 3 are much rarer than lattices of size 2 or 4. This is because as mentioned above many of the lattices have the shape of an anti-chain or crown. But the only possibility to form a lattice with 3 concepts is as a chain, which corresponds to a subset relation among the synsets and is very rare.

Table 4 shows the percentages of lattices with fewer than 6 concepts, between 6 and 45, and more than 45 concepts. Having fewer than 6 concepts is undesirable because

Table 4. The percentages of lattices with different sizes

	WN:hypern	WN: 2-1	WN: 2-2	RT: 2-1
smaller than 6	90%	91%	52%	68%
between 6 and 45	10%	9%	42%	31%
larger than 45	0%	0%	6%	1%

such lattices may not be very interesting. On the other hand, for specialised terms, proper nouns, and so on, it may be unavoidable to have a small lattice. Lattices with 6 to 45 concepts should be viewable. If 45 concepts is too large, restriction can be applied (Priss & Old, 2004), which removes most concepts that are meet- and join-irreducible and reduces the number of concepts without changing the core structure of the lattice. Lattices with more than 45 concepts are most likely not readable and require restriction.

Table 4 indicates that in the case of WordNet a good strategy might be to create plain 2-2 neighbourhoods for most words except for the 6% for which the lattices have more than 45 concepts. For such words, hypernymy or plain 2-1 neighbourhoods should be chosen, which are guaranteed to have smaller lattices. It should be noted that this does not imply that the hypernymy relation should never be used. It is possible to combine the hypernymy and plain 2-2 neighbourhoods. Also different strategies might be necessary for the different parts of speech. For RT, plain 2-2 neighbourhoods are suitable for 68% of the words, whereas plain 2-1 neighbourhoods should be used for 31% of the words. A restricted plain 2-1 neighbourhood should be used for the remaining 1%.

From an implementation viewpoint, the sizes of the lattices should be stored in a look-up table. This can be calculated while the database is off-line, so it does not matter if this is slow. We have not yet calculated the sizes for all words in WordNet and RT, but as mentioned before, the entries which have not been included in our test data tend to be phrases and compound words which tend to have smaller lattices. Thus, most likely our list of the 6% of words with large plain 2-2 neighbourhoods is a complete or nearly complete listing for WordNet. Our list of 31% of the words for RT will also be nearly complete. Our previous approach to limiting the size of the lattices with respect to our on-line RT interface has been to use heuristics derived from the number of objects and attributes of the context. Using a look-up table instead of heuristics seems to be a better approach because it is more precise while still requiring about the same amount of computational resources.

6 Conclusion

The main result of this paper is that we have extended our concept neighbourhood modelling from Roget's Thesaurus to WordNet. We have conducted a number of experiments, both with respect to looking at individual examples of words, but also obtaining a distribution of the sizes of neighbourhood lattices of different types. The results show that there are differences between parts of speech and between WordNet and RT. A feasible approach to determining in advance which type of neighbourhood to use for which word appears to be to calculate the sizes of the neighbourhood lattices once, and store

them as a look-up table in the database. We have not yet calculated these numbers for all words and all types of semantic relations but it appears that words with large lattices in one type also have large lattices in other types. Verbs of Anglo-Saxon origin tend to have the largest lattices. The data about the lattice sizes which we have calculated so far appears sufficient to prevent the construction of oversized neighbourhood lattices. Users of the on-line interface[4] can still be given some flexibility to experiment with different types of neighbourhood lattices. Only the construction of oversized lattices needs to be avoided.

It might be of linguistic interest to conduct a more detailed analysis of the data that we have collected so far. It might indicate a classification of words according to their types of neighbourhoods or offer more insights into the structure of synonymy and other semantic relations. Also, if a word behaves differently across different neighbourhoods, such anomalies might highlight interesting facts or errors in the data. But such analyses are left for future research.

Other plans for future research include: extending this research to other lexical databases; investigating the use of faster algorithms; and improving and testing the user interface of our website. A further idea is to investigate whether there are other indicators than number of concepts that can be used to determine how readable the lattices are. For example, for the same number of concepts, lattices that have about the same width as height might be more readable than lattices that are "short" and "wide". Our experimental data suggests that "tall" and "narrow" lattices are unlikely with respect to WordNet and RT. One indicator for width might be the degree of sparseness of the formal context.

References

1. Dyvik, H.: Translations as semantic mirrors: from parallel corpus to wordnet. Language and Computers 49(1), 311–326 (2004); Rodopi
2. Fellbaum, C. (ed.): WordNet - An Electronic Lexical Database. MIT Press, Cambridge (1998)
3. Hotho, A., Staab, S., Stumme, G.: Explaining text clustering results using semantic structures. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 217–228. Springer, Heidelberg (2003)
4. Kennedy, A., Szpakowicz, S.: Evaluating Roget's Thesauri. In: Proc. of ACL 2008, HLT, Columbus Ohio, USA, pp. 416–424. Association for Computational Linguistics (2008)
5. Martin, B., Eklund, P.: Applying Formal Concept Analysis to Semantic File Systems Leveraging Wordnet. In: Proceedings of the 10th Australasian Document Computing Symposium (2005)
6. Old, L.J.: The Semantic Structure of Roget's, A Whole-Language Thesaurus. PhD Dissertation. Indiana University (2003)
7. Priss, U., Old, L.J.: Modelling Lexical Databases with Formal Concept Analysis. Journal of Universal Computer Science 10(8), 967–984 (2004)

[4] The interface for RT is currently at http://www.ketlab.org.uk/roget.html. The WordNet interface will also be added to that site in the near future.

8. Priss, U., Old, L.J.: Conceptual Exploration of Semantic Mirrors. In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 21–32. Springer, Heidelberg (2005)
9. Priss, U., Old, L.J.: An application of relation algebra to lexical databases. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 388–400. Springer, Heidelberg (2006)
10. Sedelow, S., Sedelow, W.: The Concept concept. In: Proceedings of the Fifth International Conference on Computing and Information, Sudbury, Ontario, Canada, pp. 339–343 (1993)
11. Sedelow Jr., W.A.: Computer-based planning technology: an overview of inner structure analysis. In: Sixth Annual Conference on New Technology and Higher Education: Acquisition, Integration, and Utilization (1988)

A Survey of Hybrid Representations of Concept Lattices in Conceptual Knowledge Processing

Peter Eklund[1] and Jean Villerd[2]

[1] School of Information Systems and Technology
University of Wollongong, Australia
peklund@uow.edu.au
[2] LORIA – INRIA Nancy - Grand Est Research Centre
Nancy, France
jean.villerd@loria.fr

Abstract. A feature of Formal Concept Analysis is the use of the line diagram of the concept lattice to visualize a conceptual space. The line diagram is a specialized form of Hasse diagram labeled with the object extents and the attribute intents. The line diagram is usually drawn so that its rendering maximizes symmetry and minimizes edge crossings. Further the line diagram is usually layered hierarchically from top to bottom. Variations of the line diagram are frowned upon in the mathematical treatment of Formal Concept Analysis but hybrid presentations of concept lattices have practical value when used in an appropriate application context. This paper surveys previous work on using line diagrams and further explores hybrid visual representations of concept lattices. It identifies connections to other visual information techniques in data mining and information visualisation that can be used to enhance Formal Concept Analysis applications.

1 Introduction

In the last three decades Formal Concept Analysis (FCA) [30] has grown in popularity as a method for data analysis and knowledge representation. One reason is the ability to visualize an information space as a concept lattice or line diagram. Consider the formal context in Fig. 1 and its corresponding line diagram. Fig. 1 presents subtle differences from the standard diagrammatic representation of the line diagram, it includes the use of graduated color on the vertices representing the cardinality of the object extents (the vertex at the top of the lattice being darkest and the bottom node of the lattice being the lightest). This is a minor departure from the conventions of line diagram drawing and this paper explores other more radical ideas. Our purpose with this survey is to look at hybrid visual representations of concept lattices because we believe experimentation in this direction will lead to innovations that enhance the application of Conceptual Knowledge Processing (CKP) [32] for knowledge and data discovery. As well as being useful in various applications, we argue that these departures are sanctioned by Wille's vision of CKP.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 296–311, 2010.

Table 1. Methods for Conceptual Knowledge Processing as per Wille [32], in this paper we focus on those methods which touch on line diagrams of a concept lattice, namely those arrowed (→) and in bold, (line diagrams are also significantly used in conceptual scaling, methods M3.1–M.3.6)

	Conceptual Knowledge Processing Method Name	**Reference**
	Representing a Context by a Cross Table	(M1.1)
	Clarifying a Context	(M1.2)
	Reducing a Context	(M1.3)
→	**Representation of a Concept Hierarchy by a Line Diagram**	**(M1.4)**
→	**Checking a Line Diagram of a Concept Hierarchy**	**(M1.5)**
	Dualizing a Concept Hierarchy	(M1.6)
	Generating Concepts	(M2.1)
→	**Generating All Concepts Within a Line Diagram**	**(M2.2)**
	Determining All Concepts of a Context	(M2.3)
	Determining a Context from an Ordered Collection of Ideas	(M2.4)
	Conceptual Scaling of a Context	(M3.1)
	Conceptual Scaling of a Many-valued Context	(M3.2)
	Nominal Scaling of a Many-valued Context	(M3.3)
	Ordinal Scaling of a Many-valued Context	(M3.4)
	Interordinal Scaling of a Many-valued Context	(M3.5)
	Contraordinal Scaling of a Many-valued Context	(M3.6)
	Concept Classification of Objects	(M4.1)
	Many-valued Classification of Objects	(M4.2)
→	**Partitioning Attributes of a Context (Nested Line Diagrams)**	**(M5.1)**
	Atlas-Decomposition of a Concept Hierarchy	(M5.2)
	Concept Patterns in a Concept Hierarchy	(M5.3)
	Juxtaposition of Contexts with Common Object Collection	(M6.1)
	Aggregation Based on Object Families	(M6.2)
→	**TOSCANA-Aggregation of Concept Hierarchies**	**(M6.3)**
	Identifying a Concept	(M7.1)
	Identifying Concept Patterns	(M7.2)
	Determining the Attribute Implications of a Context	(M8.1)
	Determining Many-valued Attribute Dependencies	(M8.2)
	Attribute Exploration	(M9.1)
	Concept Exploration	(M9.2)
	Discovering Association Rules	(M9.3)
	Retrieval with Contexts and Concept Hierarchies	(M10.1)
→	**Retrieval with a TOSCANA-System**	**(M10.2)**
	Theory Building with Concept Hierarchies	(M11.1)
	Theory Building with TOSCANA	(M11.2)
	Conceptual Graphs Derived from Natural Language	(M12.1)
	Derivation of Judgments from Power Context Families	(M12.2)

2 Representing Line Diagrams

Because this paper concentrates on diagrams from Conceptual Knowledge Processing (CKP) we recap Wille's [32] methods and focus on methods that involve

	small	medium	large	near	far	moon(s)	no_moon
Mercury	×			×			×
Venus	×			×			×
Earth	×			×		×	
Mars	×			×		×	
Jupiter			×		×	×	
Saturn			×		×	×	
Uranus		×			×	×	
Neptune		×			×	×	
Pluto	×				×	×	

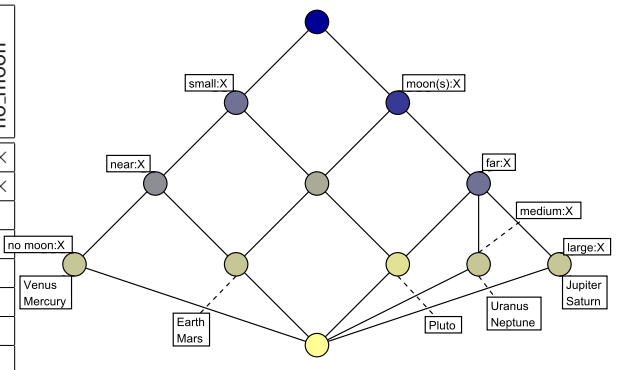


Fig. 1. A cross-table containing information about the planets and the corresponding line diagram of the concept lattice induced by the formal context

visualizing concept lattices. Wille generalizes CKP from Formal Concept Analysis (FCA), FCA being the mathematization of CKP. Table 1 lists the methods of CKP showing that the analysis framework has formalized methods, processes and algorithms. Referring to Table 1 we consider only the highlighted methods, those that relate to line diagrams of the concept lattice and elaborate each in turn. The description of the methods is abbreviated from Wille [32] with our commentary.

2.1 Representation a Concept Hierarchy by a Line Diagram (M1.4)

Concepts are represented by circles so that upward line segments between them indicate a sub-superconcept relation. Every circle representing a concept generated by an object/attribute has attached from below/above the name of that object/attribute. The labels of object and attribute names allow one to read off the extension and intension of each concept from the line diagram. Starting from any given concept, the extension/intension of a concept consists of all those objects/attributes the names of which are attached to a circle belonging to a downward/upward path of line segments [32].

2.2 Checking a Line Diagram of a Concept Hierarchy (M1.5)

A line diagram represents a concept hierarchy iff the line diagram satisfies: (1) each circle starting a downward line segment must have attached an object name; (2) each circle starting an upward line segment must have attached an attribute name; (3) an object g has an attribute m in the given context iff the names of g and m are attached to the same circle or there is an upward path of line segments from the circle with the name of g to the circle with the name of m; (4) The Basic Theorem on Concept Lattices holds: namely the lattice is complete [32].

The conditions on **(M1.4)** and **(M1.5)** relate to drawing line diagrams and are quite strict as to what constitutes a line diagram of a concept lattice. In [32], Wille describes algorithmic methods for Conceptual Knowledge Processing including an algorithmic description for generating the line diagram in **(M2.2)**. ConImp [2] was developed in the mid-80s and is the progenitor FCA program but has no capability for lattice rendering, the program is used to manipulate formal contexts and compute concept listings from which a line diagram can be then be drawn by hand. Because ConImp had no automatic mechanism for rendering line diagrams, it encouraged a craft of lattice drawing and certain conventions emerged to inform the process of drawing the line diagram of a concept lattice. Drawing a concept lattice became somewhat of a art with its own conventions and is the main reason that the methods **(M1.4)**, **(M1.5)** and **(M2.2)** from Table 1 are codified in the prescriptive way Wille describes.

2.3 Generating All Concepts within a Line Diagram: (M2.2)

First, represent the concept having the full object set as its extension by a small circle and attach (from above) the names of all attributes that apply to all these objects. Secondly, choose from the remaining attributes the extension of which are maximal, draw for each of them a circle below, link them to the parent circle by a line segment, and attach (from above) the corresponding attribute names. Then determine all intersections of the extensions of the existing concepts and represent the concepts generated by those intersections by small circles with their respective line segments representing the subconcept-superconcept-relationships. Perform analogously until all attributes are treated. Finally, attach each object name (from below) to that circle from which upward paths of line segments lead exactly to those circles with attached names of attributes applying to the respective object ([33], p.64ff.) [32].

Taken in combination **(M1.4)** **(M1.5)** and **(M2.2)** provide prescriptive detail about the graphical elements to use in the construction of the line diagram, namely small circles and line segments, labels of attributes and objects. It also codifies a top-to-bottom rendering of the conceptual hierarchy. On the other hand, the methods say nothing about layout or the length of line segments. Despite small-scale success with hand-drawing line diagrams. the process of creating and laying-out a line diagram has frustrated full automation, *"... up to now, no universal method is known for drawing well readable line diagrams representing concept hierarchies. For smaller concept hierarchies, the method of Drawing an Additive Line Diagram (see [33], p.75) often leads to well-structured line diagrams. This is the reason that quite a number of computer programs for drawing concept hierarchies use that method (e.g.* Anaconda, Cernato, ConExp, Elba*)"[32].* The result is that drawing line diagrams can only be partially automated and the diagrams improved by direct manipulation by humans.

2.4 Drawing Line Diagrams

In practice two main approaches for drawing line diagrams can be identified. The first approach is a vector-based method that uses the notion of additive line

diagram [33] and aims to optimize the interpretability of the diagram with respect to the original context. Attribute and/or object concepts – the labeled circles described in **(M1.4)** – are drawn first, determining the positions of the remaining concepts. The second approach considers the line diagram as a directed graph, and adapts existing graph drawing techniques for producing aesthetics diagrams by, among other things, minimizing edge crossings and maximizing symmetry. In this case the order relation between concepts is the only information taken into account, and all concepts are of the same importance.

Additive line diagrams approach. Ganter and Wille [33] specify an additive line diagram associated to a concept lattice $\underline{\mathfrak{B}}(G, M, I)$ with the help of two functions:

1. A representation function $\text{rep} : \underline{\mathfrak{B}}(G, M, I) \to \mathfrak{P}(X)$ that assigns to each concept a subset of a representation set X. This function must preserve the partial order of the lattice, i.e. $c_i \leqslant c_j \Leftrightarrow \text{rep}(c_i) \subseteq \text{rep}(c_j)$ for two concepts c_i and c_j. Commonly, the attribute set M is used as the representation set and then we have $\text{rep}(c) = \text{Int}(c)$.
2. A grid projection $\text{vec} : X \to \mathbb{R}^2$ assigning a real vector with a positive y coordinate to each element of X (commonly to each attribute). Then the position of a concept c on the plane follows $\text{pos}(c) = n + \sum_{x \in \text{rep}(c)} \text{vec}(x)$ where the vector n is used to shift the location of the lattice on the display.

Additive line diagrams have the advantage of being tunable through the choice of the representation set, and the technique produces a great number of parallel edges, that in turn improves readability.

Force-directed approach. A common graph drawing technique used in the second approach is the force-directed method. The basic idea is to use a physical analogy to model the graph drawing problem. A system of forces is applied to the vertices, which move until a configuration that minimizes the potential energy of the system is reached. An advantage of this method is that it often reveals symmetries existing in the graph. A well-known implementation of this method is Eades' *Spring embedder* model [12], in which two types of forces operate. An attraction force, which makes edges act as springs, that follows a logarithmic force law; and a repulsion force which makes vertices act like positive electrical point charges that repel each other to avoid overlapping, following an inverse-square force law.

In particular, following these ideas Freese [19] proposed a force-directed method for drawing lattices. In order to preserve the top-to-bottom rendering, the y coordinates of the vertices/concepts are computed w.r.t. their depth in the lattice, using a layering approach. Then the forces only impact the x coordinates. The attraction force acts between comparable concepts, so that a concept remains close to its predecessors, and the repulsion force acts between incomparable concepts of the same layer in order to avoid overlapping. Cole [4,6] combined the force-directed and the additive approaches and introduced several metrics that quantify the *goodness* of a diagram layout.

In the layout strategies mentioned above, no semantics is attached to the length of the edges nor to the distances between concepts in general, and nothing is said about this in **(M1.4) (M1.5)** and **(M2.2)**. However, one can anticipate a layout strategy where the distances between vertices reflect inversely on the similarity between concepts. This can be done using Kamada and Kamai's force directed approach [22] that modifies the *Spring embedder* model by removing the repulsion force and extending the attraction force to all pairs of vertices, i.e. a spring is attached to all pairs of vertices, whenever an edge exists between the two vertices or not. The attraction force follows Hooke's law, so that the force exerted on the vertices is proportional to the difference between the spring's rest length and the actual distance between the vertices. Each iteration refines the layout of the graph, until the system converges to a state of minimal energy. Although the system may fall into a local optimum, the final layout is intended to provide a faithful representation of the distances between vertices. This approach can also be applied in Multidimensional scaling (MDS) where the layout of high-dimensional objects in a low-dimensional projection is the goal.

Several similarity measures between formal concepts have been proposed, including [23,27,21] and some have been applied and studied in [8]. These are based on the symmetric difference between sets and differ in the type of sets involved (extents and/or intents), and in their sensitivity (local dissimilarity or general similarity involving all the objects and/or attributes of the context). Note that all these metrics are distances, a nice property for layout. To our knowledge, only Hannan and Pogel [21] used distance for the purposes of layout as is shown in Fig. 2. In their work, the distance between two concepts c_i and c_j is the cardinal of the symmetric difference between respective extents, i.e. $d(c_i, c_j) = |\text{Ext}(c_i) \triangle \text{Ext}(c_j)| = |(\text{Ext}(c_i) \setminus \text{Ext}(c_j)) \cup (\text{Ext}(c_j) \setminus \text{Ext}(c_i))|$ with the aim to improve the layout of additive line diagrams by applying a force-directed post-processing based on this distance. The final layout – an example of which is Fig. 2 – shows a diagram in which, the more concepts share common objects the closer they appear. Thus, considering two concepts drawn near one another, the association rules between their respective intents may be of high confidence, suggesting visual discovery of *almost exact* rules from layout.

The attributes (resp. contexts) considered so far are binary (i.e. one-valued contexts). In order to handle many-valued contexts containing non-binary attributes such as *gender*, *colour*, *grade* or *age*, a discretization process called *conceptual scaling* is used **(M3.2)**. The core idea is to transform the many-valued context into a *derived* one-valued context according to a set of rules that depends both on the nature of the many-valued attributes (nominal, ordinal or numerical) and on some potential background knowledge about relations between their values. The *conceptual scaling* mechanism itself is out of the scope of the present paper. However since it results in the creation of several new binary attributes for each many-valued attribute, the number of concepts extracted from the derived one-valued context may grow exponentially (in the worst case, $2^{\min(|G|,|M|)}$ concepts are extracted from a context with $|G|$ objects and $|M|$ attributes), leading to unreadable line diagrams. The following section

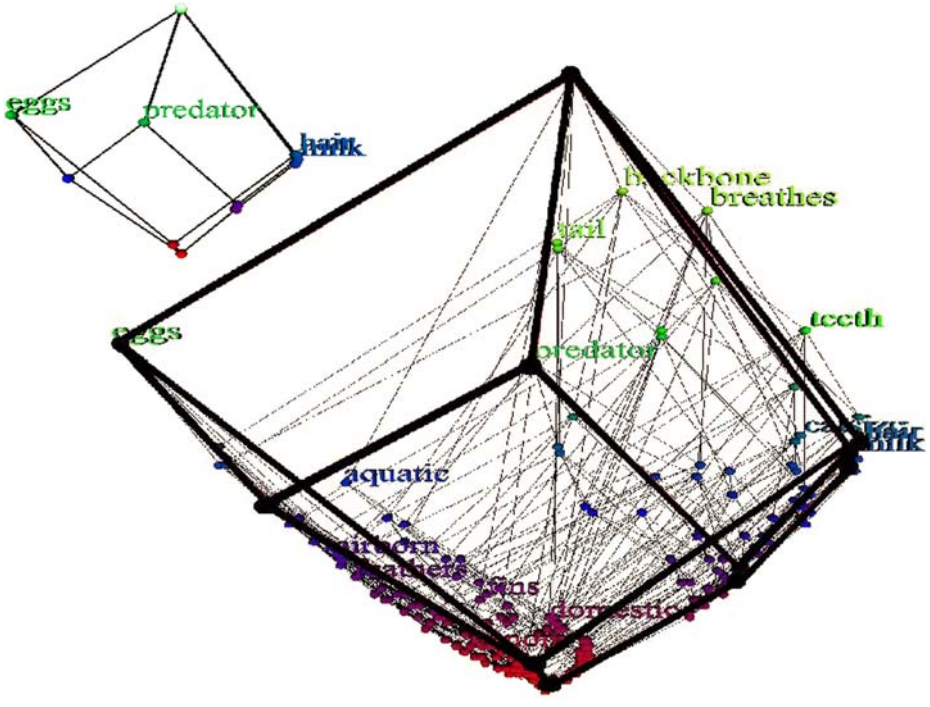


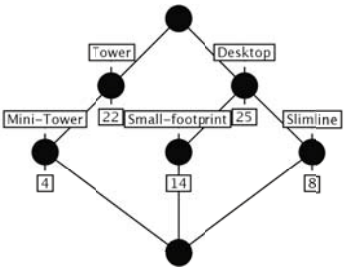
Fig. 2. The concept lattice of the UCI zoological dataset of 101 animals and 15 binary attributes. This lattice has 238 concepts. The concepts "milk" and "hair" are drawn very near each other (the attribute labels overlap – extreme right), while the concept "eggs" is repulsed from the pair – left. The rules "milk"→"hair" and "hair"→"milk", have confidence 95.1% and 90.6% resp. Visual clustering of "milk" and "hair" suggest they are minor variations of one another (re-printed with permission).

deals with nested line diagrams, an alternative technique to represent and study large contexts by partitioning the attributes.

2.5 Partition Attributes of a Context (Nested line diagrams (M5.1))

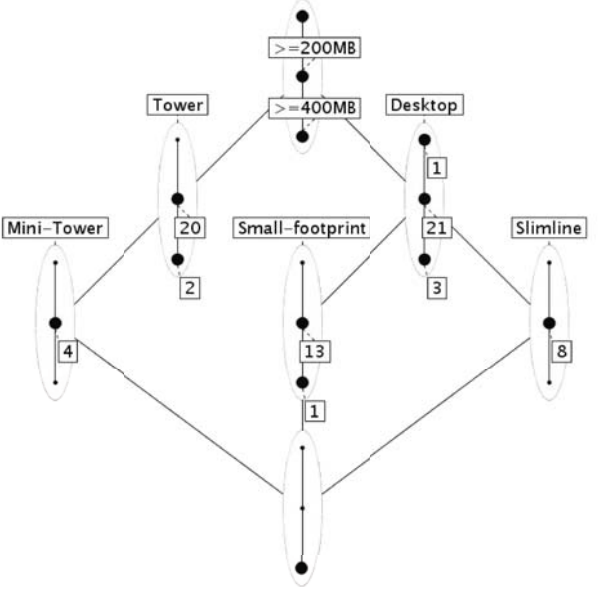
A concept hierarchy can be visualized as a nested line diagram constructed as follows (cf. [33], p.75ff.): First, line diagrams of the concept lattices of the subcontexts are prepared and ordered in a sequence of subcontexts. Then, the line diagram being second in the sequence is copied into each circle of the line diagram; next, the line diagram being third in the sequence is copied into each circle of the line diagram second in the sequence and so on [32].

In TOSCANAJ [1] histograms were used to represent object extents and clustergrams used to display attributes. A key question arising from **(M5.1)** is whether is makes sense to embed a visualization of anything other than a line diagram within a nested line diagram.



(a) *chassis type* line diagram

(b) *HDD capacity* line diagram



(c) resulting nested line diagram

Fig. 3. A nested line diagram (c) built the from two subcontexts *chassis types* (a) and *HDD storage* (b). Numbers represent extent cardinality

Nested line diagrams are particularly suited to contexts derived from many-valued contexts. Fig. 3 shows the building of a nested line diagram from a many-valued context containing computers as objects and two many-valued attributes *chassis type* and *HDD capacity* (computed using TOSCANAJ). Each of these attributes has been scaled into a set of binary attributes that constitute two subcontexts. Following the above instructions, the line diagrams of the two subcontexts are drawn (see Fig. 3(a) and 3(b)), and then the second is copied into each circle of the first (Fig. 3(c)). One can easily observe the distribution of HDD capacities along with the type of chassis. However, two limitations arise: (i) combining more than two many-valued attributes may result in complex nested line diagrams, useful for deep analysis but not suitable for a first-glance navigation; (ii) the scaling pre-processing inevitably results in a loss of precision concerning

numerical attributes. Hence, values for the numerical attribute *HDD capacity* have been gathered into two binary attributes, namely $\geqslant$ 200MB and $\geqslant$ 400MB using an ordinal scale **(M3.4)**. When the granularity is modified in this way, the entire nested line diagram must be redrawn. Hybrid solutions to address both these limitations will presented in Section 2.7 below.

2.6 Toscana-Aggregation of Concept Hierarchies (M6.3)

The idea of a TOSCANA-aggregation is to view a related system of concept hierarchies metaphorically as a conceptual landscape of knowledge that can be explored by a purpose-oriented combination and inspection of suitable selections of the given concept hierarchies [32].

Groh [20] was the first to experiment with visual abstraction, using an abbreviated form of the line diagram for displaying flight routes in a network as reported in [15]. Further, the conceptual landscape idea [31] was explored extensively by Ducrou [9] and the results were a number of hybrid TOSCANA-systems. Wille's landscape metaphor can therefore be said to have significantly encouraged experimentation with visualization and line diagram abstraction.

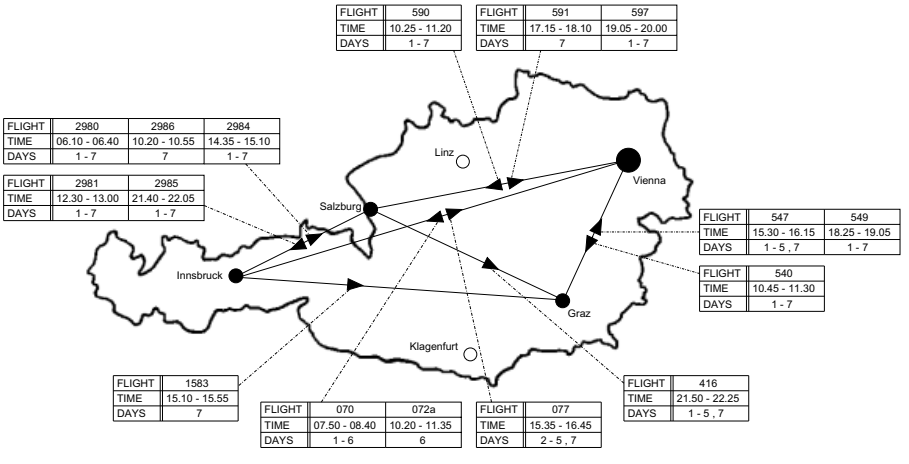


Fig. 4. A diagram showing a representation of a relational power context family modeling a network of airline flights in Austria. The network is depicted as a directed graph and the labels on line segments as relations. The digraph is placed within a backdrop map of Austria with the position of the vertices corresponding to the geo-coordinates of Austrian cities.

2.7 Retrieval with a Toscana-System (M10.2)

The term TOSCANA-system appears to refer to a particular software framework for Formal Concept Analysis, however it is intended by Stumme et al. [29] to mean a general class of Formal Concept Analysis systems that formalize a dataset into a context and to provide different points of view on the data by assisting

the user in building and applying different conceptual scales. With this idea in mind we now look how various FCA-applications have dealt with visualization and retrieval. Conceptual knowledge retrieval is often a process where humans search for something with only a vague information need. Therefore humans learn step by step how to specify what they are searching for. Line diagrams of the activated scales are shown to the user who learns by inspecting them on how to act further [32]. We show how the navigation facilities of line diagram can be enhanced through their combination with non-FCA visualization techniques.

Line diagrams as a support for navigation. The powerful classification ability of FCA has found many applications in information retrieval. Some of them have been listed by [26]. Carpineto and Romano [3] argue that, in addition to their classification behaviors for information retrieval tasks, concept lattices can also support an integration of querying and browsing by allowing users to navigate into search results. Nowadays, several Formal Concept Analysis-based applications like CREDO [3], MAILSLEUTH [5,7] or SEARCHSLEUTH [8] have been developed. Upstream research has studied the understandability of a lattice representation by novice users [13,11]. A program called IMAGESLEUTH [10,14] further proposes an interactive Formal Concept Analysis-based image retrieval system in which subjacent lattices are hidden. Users do not interact with an explicit representation of a lattice. They navigate from one concept to another by adding or removing terms suggested by the system. The same design approach is followed in SEARCHSLEUTH [8] and also in [18]. This ensures progressive navigation within the lattice (Wille's method **10.2**).

An hybrid approach to enhance readability. During the usability review of MAILSLEUTH in 2003, [13] encountered users who felt that the drawing conventions for a lattice diagram were no different from a graph in mathematics.

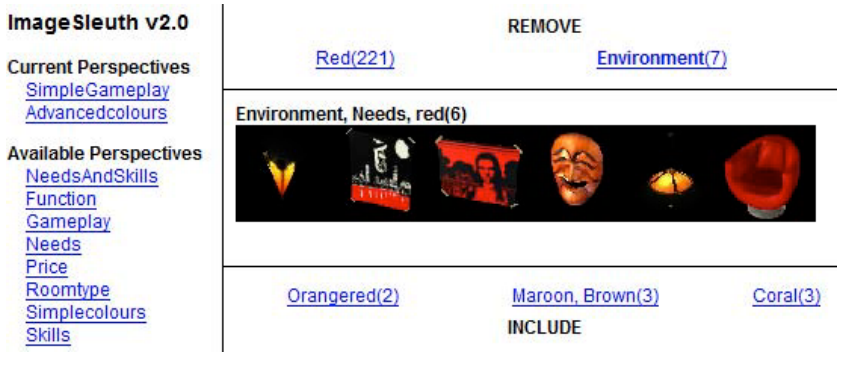


Fig. 5. ImageSleuth: the interface presents only the extent of the current concept as thumbnails and generalizations/specializations by removal/addition of attributes to reach the upper and lower neighbors (shown to the top/bottom of the thumbnails). Pre-defined scales (perspectives) are displayed on the left.

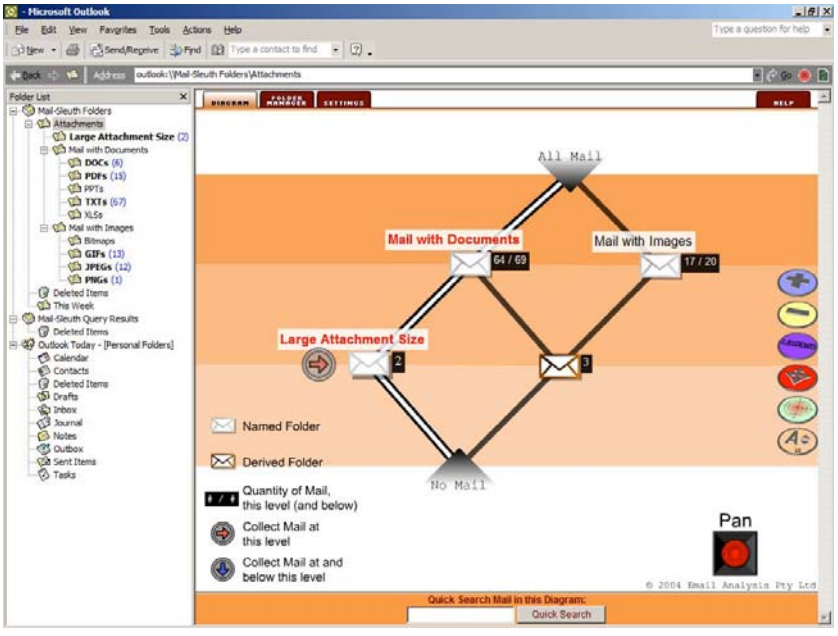


Fig. 6. A screenshot of the MailSleuth program

What makes this a lattice diagram and not a graph? In Hasse diagrams, line segments (represent the cover relation) are unlabelled. It is well understood in mathematics that a partially ordered set is transitive, reflexive and antisymmetric and so to simplify the drawing of an ordered set (via its cover relation) the reflexive and transitive edges are removed, and the directional arrows of the relation are dropped. It is understood by convention in Mathematics that the Hasse diagram is hierarchical with the edges pointing upward. In other words, if $x < y$ in the partially ordered set then x appears at a clearly lower point than y in the diagram **(M1.5)**. To insinuate structure to novices, the idea of a line diagram with shaded layers was introduced in MailSleuth. The principle is simple, with dark at the top and light at the bottom; the shades progressively lighter as one moves from one level to the next (shown in Fig. 6). The top and bottom elements of the lattice have also been replaced with icons indicating All Mail and No Mail (when the bottom element is the empty set of objects).

Shading does not interfere with Formal Concept Analysis diagrammatic conventions because it is a backdrop to the line diagram. It can also be turned off if the line diagram is to be embedded in a printed document. However, the interaction of the layout algorithm and background layer shading fails (background layers are not aligned) in line diagrams with high dimensionality. On the other hand is possible to use the alignment of the background layers to guide the manual layout process. Once layer shading is used, users are better able to explain (and read) the line diagram from top-to-bottom and bottom-to-top.

Because MailSleuth deals with objects that are emails, it was natural to replace vertices with a literal iconic representation relevant to the domain. In the case where Derived Folders are unrealised, no vertex is drawn at all. Where data is present, an envelope is used, the envelopes animate by "appearing to lift" on rollover with drop shadowing. This helps suggest that vertices in the line diagram can be moved and therefore manually adjusted by the user. Edge highlighting has been used to emphasise relationships in line diagrams in both ToscanaJ and in MailSleuth. This idea is used as a method to orient the current vertex in the overall line diagram so that relationships can be identified.

In many Toscana-systems the set of objects in the extent is often replaced with a number representing the cardinality of the extent (and/or the contingent), this can extended to allow the edges of the line diagram to be labelled with the ratio of object counts to approximate the idea of support in data mining.

Several strategies can be applied to reduce the number of vertices displayed to avoid visual overloading. Iceberg lattices [28] and alpha lattices [25] are smaller concept lattices retaining significant concepts that respect a threshold criterion based on the extent cardinality. Another lattice reduction strategy is decomposing the overall lattice into smaller sublattices, as done in ImageSleuth [10,14], with this idea a partition of term space is achieved by a domain expert, attribute filtering or object zooming.

2.8 A Hybrid Approach to Handle Numerical Attribute

Considering a many-valued context, we have seen that nested line diagrams encounter two major limitations. First, combining more than two many-valued attributes makes the nested line diagram unreadable. Secondly, while no information is lost when building a line diagram from a binary context, the binarization of numerical attributes leads to a loss of precision. An hybrid solution to address these limitations is to partition the context into binary and nominal attributes on the one hand, and ordinal and numerical attributes on the other hand. The binary/nominal line diagram is drawn but its circles are not filled with the ordinal/numerical line diagram. They are filled with a visualization that represents proximities between objects w.r.t. ordinal and numerical attributes. For instance, the circle labeled *Desktop* in Fig. 3(c) would contain 25 points that correspond to the 25 objects in the extent of this concept. The points are drawn in such a way that the distance between two points reflects the distance between the two corresponding objects w.r.t. the numerical attribute *HDD capacity*. These embedded visualizations can be drawn using a force-directed Multidimensional scaling technique [24], which takes as input the Euclidian distances between objects w.r.t. a given set of numerical attributes, and produces 2D embedding of the objects. This solution addresses both the above limitations since (i) the Euclidian distances can be computed w.r.t. any number of attributes; (ii) no scaling pre-processing is required. Note that the previous scaling pre-processing resulted in a predefined clustering of objects, whereas clusters of objects that can appear in embedded visualizations may not be expected, providing new insights into the data. However, this solution is not accurate when the number

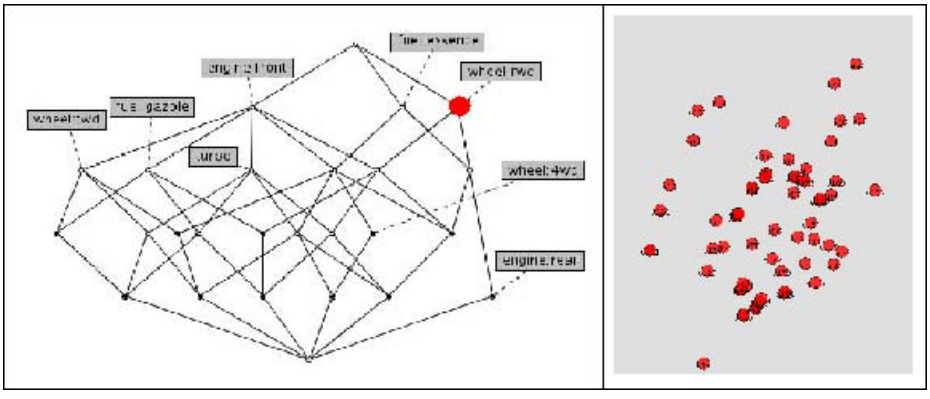


Fig. 7. Hybrid representation associated to a many-valued context. The line diagram left is built from the subcontext containing binary and nominal attributes. The objects contained in the extent of the selected concept are shown on the right by an MDS embedding that reflects their proximities w.r.t. numerical attributes.

of objects and binary/nominal attributes grows. Hence, embedded visualization may become overcrowded and too small to be useful. A solution is to follow the *overview + detail* paradigm that splits the screen into two views (see Fig. 7). The overview shows the line diagram built from binary and nominal attributes, while the detailed view shows the embedded visualization of the selected vertex in the overview. The detailed view acts as a kind of magnifying glass on the content (the extent) of the selected concept.

2.9 An Hybrid Approach to Provide Insights Concerning Navigation Costs Regarding a Concept's Neighbors

A final idea is to combine tag clouds with Formal Concept Analysis. This is shown in Fig. 8. This shows a tagging and annotation system called the Virtual Museum of the Pacific [17,16]. Here, any attributes that lead to the upper neighbors are shown above the images, in this case `male, melanesia` and `PNG` and attributes (when added) that lead to lower neighbors in the concept lattice are shown below the images, in this example `fishing, container, spiritual` and `magic`. The size of the attributes is determined by the size the extent of the concepts that they lead to so the position and font size of upper and lower neighbors is conditioned by the extent sizes of the formal concepts they lead to. For example, in this case `fishing` is shown larger than `container, spiritual` and `magic` since it is a smaller specialization while `container` and `magic` are the same size and so have the same number of objects in their extents. The result is two tag clouds, one for the attributes leading to upper neighbors and another for lower neighbors.

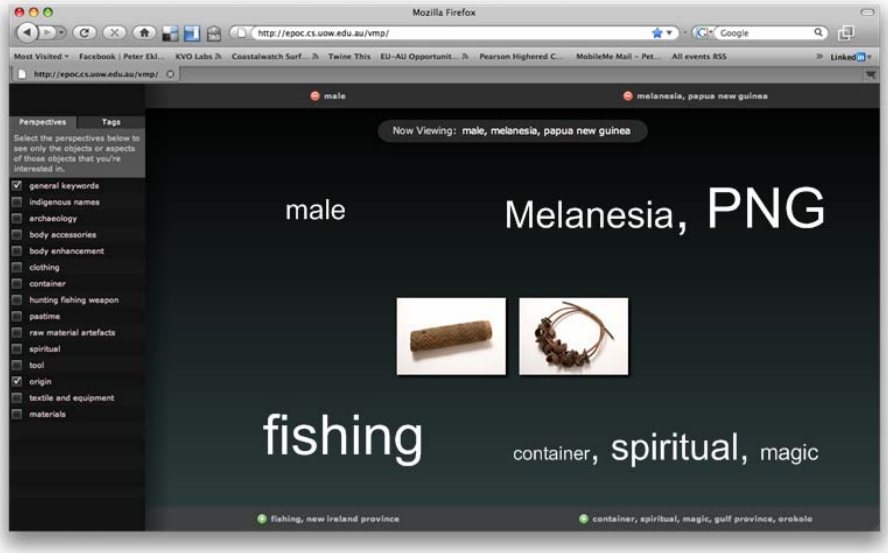


Fig. 8. A screenshot of the Virtual Museum of the Pacific program showing upper and lower neighbors of a formal concept as a tag cloud. The larger the lower neighbour, the larger the extent of the resulting formal concept being navigated to (minimal change). The larger the upper neighbor the greater the generalisation (maximal change).

3 Conclusion

This paper is a survey of visualization using line diagrams of concept lattices but with specific purpose. We argue that in the strict understanding of Conceptual Knowledge Processing variations of the standard techniques of line diagram drawing are supported. Further, that in various analytical contexts for Knowledge and Data Discovery, these variations of useful. Our examples cover the coloring of vertices to present the cardinality of formal concept extents; the placement of vertices of the line diagram to reflect association rule confidence; the use of icons for vertices, layer shading and abstractions of the line diagram that conform to Wille's landscape metaphors; and the use of iceberg lattices to minimize visual clutter and using similarity measures to condition the length of edges. Further, we have demonstrated the utility of showing only the conceptual neighborhood of the concept lattice and finally mixing this idea with the traditional idea of an attribute tag-cloud.

References

1. Becker, P., Hereth Correia, J.: The ToscanaJ suite for implementing Conceptual Information Systems. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 324–348. Springer, Heidelberg (2005)

2. Burmeister, P.: Formal concept analysis with conimp: Introduction to the basic features. Technical report, TU-Darmstadt (1996), http://www.mathematik.tu-darmstadt.de/~burmeister
3. Carpineto, C., Romano, G.: Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. Journal of Universal Computer Science 10(8), 985–1013 (2004)
4. Cole, R.: Automatic layout of concept lattices using force directed placement and genetic algorithms. In: Proc. of the 23th Australiasian Computer Science Conference, pp. 47–53. IEEE Computer Society, Los Alamitos (2000)
5. Cole, R., Eklund, P., Stumme, G.: CEM — A Program for Visualization and Discovery in Email. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 367–374. Springer, Heidelberg (2000)
6. Cole, R.J., Ducrou, J., Eklund, P.: Automated layout of small lattices using layer diagrams. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3874, pp. 291–305. Springer, Heidelberg (2006)
7. Cole, R.J., Eklund, P., Stumme, G.: CEM - a program for visualization and discovery in email. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 367–374. Springer, Heidelberg (2000)
8. Dau, F., Ducrou, J., Eklund, P.: Concept Similarity and Related Categories in SearchSleuth. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 255–268. Springer, Heidelberg (2008)
9. Ducrou, J.: Design for Conceptual Knowledge Processing: Case Studies in Applied Formal Concept Analysis. PhD thesis, School of Information Technology and Computer Science, The University of Wollongong (2007)
10. Ducrou, J., Eklund, P.: An intelligent user interface for browsing and search MPEG-7 images using concept lattices. Int. Journal of Foundations of Computer Science 19(2), 359–381 (2008)
11. Ducrou, J., Eklund, P.: Faceted document navigation using conceptual structures. In: Hitzler, P., Schärf, H. (eds.) Conceptual Structures in Practice, pp. 251–278. CRC Press, Boca Raton (2009)
12. Eades, P.: A heuristic for graph drawing. Congressus Numerantium 42, 149–160 (1984)
13. Eklund, P., Ducrou, J., Brawn, P.: Concept lattices for information visualization: Can novices read line diagrams. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 57–73. Springer, Heidelberg (2004)
14. Eklund, P., Ducrou, J., Wilson, T.: An intelligent user interface for browsing and search MPEG-7 images using concept lattices. In: Yahia, S.B., Nguifo, E.M., Belohlavek, R. (eds.) CLA 2006. LNCS (LNAI), vol. 4923, pp. 1–21. Springer, Heidelberg (2008)
15. Eklund, P., Groh, B., Stumme, G., Wille, R.: A contextual-logic extension of toscana. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 453–467. Springer, Heidelberg (2000)
16. Eklund, P., Wray, T., Ducrou, J.: Web services and digital ecosystem support using formal concept analysis. In: Spyratos, N. (ed.) The International ACM Conference on Management of Emergent Digital EcoSystems (MEDES 2009). ACM Press, New York (in press, 2009)
17. Eklund, P., Wray, T., Goodall, P., Bunt, B., Lawson, A., Christidis, L., Daniels, V., Van Olffen, M.: Designing the Digital Ecosystem of the Virtual Museum of the Pacific. In: 3rd IEEE International Conference on Digital Ecosystems and Technologies, pp. 805–811. IEEE Press, Los Alamitos (2009)

18. Ferre, S.: Camelis: a logical information system to organize and browse a collection of documents. Int. J. General Systems 38(4) (2009)
19. Freese, R.: Automated lattice drawing. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 112–127. Springer, Heidelberg (2004)
20. Groh, B.: A Contextual-Logic Framework Based on Relational Power Context Families. PhD thesis, School of Information Technology, Griffith University (2002)
21. Hannan, T., Pogel, A.: Spring-based lattice drawing highlighting conceptual similarity. In: Proceedings of the International Conference on Formal Concept Analysis, ICFCA 2006, Berlin. LNCS, vol. 3974, pp. 264–279. Springer, Heidelberg (2006)
22. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Information Processing Letters 31(1), 7–15 (1989)
23. Lengnink, K.: Ahnlichkeit als Distanz in Begriffsverbänden. In: Wille, R., Stumme, G. (eds.) Begriffliche Wissensverarbeitung: Methoden und Anwendungen, pp. 57–71. Springer, Heidelberg (2001)
24. Morrison, A., Chalmers, M.: Improving Hybrid MDS with Pivot-Based Searching. In: Proceedings of the 2003 IEEE Symposium on Information Visualization (Info Vis 2003), pp. 85–90 (2003)
25. Pernelle, N., Ventos, V., Soldano, H.: ZooM: Alpha Galois Lattices for Conceptual Clustering. In: Proc. of the Managing Specialization/Generalization Hierarchies (MASPEGHI) Workshop (2003)
26. Priss, U.: Formal concept analysis in information science. Annual Review of Information Science and Technology 40, 521–543 (2006)
27. Saquer, J., Deogun, J.S.: Concept aproximations based on rough sets and similarity measures. Int. J. Appl. Math. Comput. Sci. 11, 655–674 (2001)
28. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with Titanic. Data & Knowledge Engineering 42(2), 189–222 (2002)
29. Stumme, G., Wille, R., Wille, U.: Conceptual knowledge discovery in databases using formal concept analysis methods. In: Żytkow, J.M. (ed.) PKDD 1998. LNCS, vol. 1510, pp. 450–458. Springer, Heidelberg (1998)
30. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered Sets, pp. 445–470. Reidel (1982)
31. Wille, R.: Conceptual landscapes of knowledge: A pragmatic paradigm for knowledge processing. In: Gaul, W., Locarek-Junge, H. (eds.) Classification in the Information Age, pp. 344–356. Springer, Heidelberg (1999)
32. Wille, R.: Methods of conceptual knowledge processing. In: Missaoui, R., Schmidt, J. (eds.) ICFCA 2006. LNCS (LNAI), vol. 3874, pp. 1–29. Springer, Heidelberg (2006)
33. Wille, R., Ganter, B.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)

TWO BASIC ALGORITHMS IN CONCEPT ANALYSIS*

Bernhard Ganter

Preprint-Nr. 831 Juni 1984

Abstract: We describe two algorithms for closure systems. The purpose of the first is to produce all closed sets of a given closure operator. The second constructs a minimal family of implications for the "logic" of a closure system. These algorithms then are applied to problems in concept analysis: Determining all concepts of a given context and describing the dependencies between attributes. The problem of finding all concepts is equivalent, e.g., to finding all maximal complete bipartite subgraphs of a bipartite graph.

* A version of this preprint was published in German as part of the book "Beiträge zur Begriffsanalyse" (Ganter, Wille and Wolff, eds., BI-Wissenschaftsverlag 1987). Computer programs for FCA were already in use in 1984. The new algorithm was not only more efficient, it could also be used to compute the "canonical base" of implications in a finite formal context that had recently been discovered by J.L. Guigues and V. Duquenne.

L. Kwuida and B. Sertkaya (Eds.): ICFCA 2010, LNAI 5986, pp. 312–340, 2010.

§1 CLOSURE SYSTEMS

Let 2^M denote the power set of a set M. By a CLOSURE OPERATOR we mean a mapping

$$\bar{}: 2^M \to 2^M$$

which is extensive, monotone, and idempotent, i.e. which satisfies for all $A,B \subseteq M$

a) $A \subseteq \bar{A}$

b) $A \subseteq B \Rightarrow \bar{A} \subseteq \bar{B}$

c) $\bar{\bar{A}} = \bar{A}$.

A set X is CLOSED iff $X = \bar{X}$. The collection of all closed sets of some closure operator is called a CLOSURE SYSTEM . It is easy to see that a family $\mathfrak{F} \subseteq 2^M$ is a closure system if and only if $M \in \mathfrak{F}$ and $\mathfrak{F}$ is closed under arbitrary intersections. The corresponding closure operator then is given by

$$A \to \bar{A} := \bigcap \{C \in \mathfrak{F} \mid A \subseteq C\} .$$

There may be as many as $2^{|M|}$ closed sets. Thus, any alorithm computing all closed sets will require a computing time exponential in $|M|$, even if the time for computing $\bar{X}$ from X is neglected. Nevertheless, some algorithms are slower than others, and the slow ones seem to be more popular.

In this paragraph we describe a simple algorithm which requires

per closed set

- at most $|M|$ steps, each of which consists of
- a single application of the closure operator, plus
- some elementary set manipulations (of complexity $O(|M|)$).

Moreover, the total number of steps never exceeds $2^{|M|}$.

We shall adapt a method which has proven successful for many similar problems. Think of the subsets of M as being lexicographically ordered, and suppose that we have an algorithm that produces from a given closed set the lexicographically next closed set. We can scan through all closed sets by repeated applications of this algorithm: We start with the lexicographically first set (which is the closure of the empty set) and apply the algorithm again and again until we reach the lexicographically last set (which is M).

There is, however, some ambiguity to the term "lexicographic order", when applied to sets. To be more precise, we describe formally a linear ordering of the subsets of M, to which we shall refer as the LECTIC ORDER. We assume that M itself is linearly ordered, w.l.o.g. $M = \{1,2,\ldots,m\}$.

1.1. DEFINITION: For $A,B \subseteq \{1,2,\ldots,m\}$, define

$$A < B \;:\Leftrightarrow\; \exists_{i \in B \setminus A}\; A \cap \{1,2,\ldots,i-1\} = B \cap \{1,2,\ldots,i-1\} .$$

We read $A < B$ as "A is lectically smaller than B" . Verbally, $A \leq B$ iff the smallest element in which A and B differ belongs to B. It is quite obvious that this defines a linear order. Note that $A \subseteq B$ implies $A \leq B$. We should mention that the lectic order, as it is defined above, is the same as the lexicographic order of the characteristic vectors. (As usual, the characteristic vector of a set $A \subseteq M$ is the binary tuple $a_1 a_2 \ldots a_m$, where $a_i = 1$ iff $i \in A$.) It is immediate from 1.1 that $A < B \Leftrightarrow$ the characteristic vector of A is lexicographically smaller than the characteristic vector of B.

Our next aim is to characterize the lectic successor of a given set $A \subseteq M$, M finite. We prepare by introducing two notations and by listing, without proof, some easy facts:

1.2 DEFINITIONS: For $A,B \subseteq M$, $i \in M$, define

1) $A <_i B :\Leftrightarrow i \in B \setminus A$ and $A \cap \{1,2,\ldots,i-1\} = B \cap \{1,2,\ldots,i-1\}$

2) $A \oplus i := \overline{A \cap \{1,2,\ldots,i-1\} \cup \{i\}}$

1.3 PROPOSITION:

1) $A < B$ iff $A <_i B$ for some $i \in \{1,2,\ldots,m\}$.

2) If $A <_i B$, $A <_j C$, and $i < j$, then $C <_i B$.

3) $i \notin A \Rightarrow A < A \oplus i$.

4) If $A <_i B$ for some closed set B, then $A \oplus i \subseteq B$, thus $A \oplus i \leq B$.

5) If $A <_i B$ for some closed set B, then $A <_i A \oplus i$.

1.4 LEMMA:

With respect to the lectic order, the smallest closed set greater than a given set $A \subseteq M$ is

$$A \oplus i ,$$

where i is the largest element of M satisfying

$$A <_i A \oplus i$$

Proof: Let A^+ be the lectic successor of A. We have to show that A is of the form described in 1.4.
Since $A < A^+$ we have $A <_i A^+$ for some $i \in M$. By 1.3.5 we conclude that $A <_i A \oplus i$, and from 1.3.4 we get that $A \oplus i \leq A^+$, thus $A^+ = A \oplus i$. It remains to prove that this i is as large as possible. Suppose $A <_j A \oplus j$ for some $j \neq i$. Since $A \oplus i = A^+ < A \oplus j$ we can use 1.3.2 to deduce $j < i$.

It is not difficult to translate 1.4 into an algorithm:

1.5 ALGORITHM to compute the successor A_{next} of a given set A with respect to the lectic order.

(A) For $i = m, m-1, \ldots, 2, 1$ step -1

(B) If $A <_i A \oplus i$ then return with $A_{next} \leftarrow A \oplus i$

(C) Next i

(D) Stop: No successor (input was $A = M$).

A more detailed version of step (B) is:

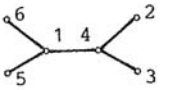
(B1) If $i \in A$ then goto (C)

(B2) $A_{next} \leftarrow \overline{A \cap \{1,2,\ldots,i-1\} \cup \{i\}}$

(B3) If there is an element $j \in A_{next}$ such that $j < i$ and $j \notin A$ then goto (C)

(B4) Return

1.6 EXAMPLE:

The connected subgraphs of the tree 

, lectically ordered:

Vertex set	characteristic vector
{∅}	0 0 0 0 0 0
{6}	0 0 0 0 0 1
{5}	0 0 0 0 1 0
{4}	0 0 0 1 0 0
{3}	0 0 1 0 0 0
{3,4}	0 0 1 1 0 0
{2}	0 1 0 0 0 0
{2,4}	0 1 0 1 0 0
{2,3,4}	0 1 1 1 0 0
{1}	1 0 0 0 0 0
{1,6}	1 0 0 0 0 1
{1,5}	1 0 0 0 1 0
{1,5,6}	1 0 0 0 1 1
{1,4}	1 0 0 1 0 0
{1,4,6}	1 0 0 1 0 1
{1,4,5}	1 0 0 1 1 0
{1,4,5,6}	1 0 0 1 1 1
{1,3,4}	1 0 1 1 0 0
{1,3,4,6}	1 0 1 1 0 1
{1,3,4,5}	1 0 1 1 1 0
{1,3,4,5,6}	1 0 1 1 1 1
{1,2,4}	1 1 0 1 0 0
{1,2,4,6}	1 1 0 1 0 1
{1,2,4,5}	1 1 0 1 1 0
{1,2,4,5,6}	1 1 0 1 1 1
{1,2,3,4}	1 1 1 1 0 0
{1,2,3,4,6}	1 1 1 1 0 1
{1,2,3,4,5}	1 1 1 1 1 0
{1,2,3,4,5,6}	1 1 1 1 1 1

The rest of this paragraph is devoted to improving this algorithm. The reader who is not interested in actually programming this procedure should skip the following pages (and continue with §2!) The notation of the closed sets by means of their characteristic vectors, as in the above example, makes a useful property of the lectic order quite conspicuous: Those characteristic vectors which start with some specified tuple come one after another. The algorithm therefore can easily be modified to generated only such closed sets which contain certain specified elements and avoid certain others.

A closer inspection of 1.5 shows that the closure operator itself was used only in a special instance, namely to compute $A \oplus i$. In some cases this can simplify the computation, in particular if the closure of A is already known and can be used. Taking this into consideration we formulate a refined version of the algorithm.

1.7 ALGORITHM (modified version of 1.5) :

Input/Output:

A number $r \geq 0$,

a list $i_0 < i_1 < \ldots < i_r$ of elements of M,

a list $A_0 \subset A_1 \subset \ldots \subset A_r$ $(=A)$ of closed sets,

satisfying the following conditions:

$i_0 = 0$

$A_0 = \overline{\emptyset}$

$A_j = A_{j-1} \oplus i_j$ for $j = 1,\ldots,r$,

$A_{j-1} <_{i_j} A_j$ for $j = 1,\ldots,r$.

Algorithm:

(A) For $i = m,m-1,\ldots,1$ step -1

(B) If $i \in A_r$ then goto (G)

(C) If $i < i_r$ then $r \leftarrow r-1$; goto (C)

(D) $i_{r+1} \leftarrow i$; $A_{r+1} \leftarrow \overline{A_r \cup \{i\}}$

(E) If $A_{r+1} \setminus A_r$ contains an element $< i$ then goto (G)

(F) $r \leftarrow r+1$; return

(G) Next i

(H) Stop: No successor (input was $A_r = M$).

Initialization (when all closed sets are to be produced): The algorithm is called first with input $r = 0$, $i_o = 0$, $A_o = \emptyset$. Later, the output of the previous run is used as the new input. If this is done then A_r runs through all closed sets (Note that $\overline{\emptyset}$, the first input, never occurs as an output.).

No doubt, the rather intricate data list used as input/output-list requires some explanation. Perhaps it is helpful to see an example first:

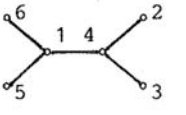


figure 1

1.8 EXAMPLE: Again we consider the subtrees of the tree in the figure. We use 1.7 to determine the lectic successor of the subtree $\{2,4\}$.

If we choose $r = 2$

$$i_o = 0,\ i_1 = 2,\ i_2 = 4$$
$$A_o = \emptyset,\ A_1 = \{2\},\ A_2 = \{2,4\}$$

then, since $A_1 = A_o \oplus i_1$

$A_2 = A_1 \oplus i_2$

$A_o <_2 \{2\}$

$A_1 <_4 \{2,4\}$,

this list is suitable as an input for 1.7. Here is what the algorithm does:

(A) $i \leftarrow 6$

(B) —

(C) —

(D) $A_3 \leftarrow \overline{\{2,4\} \cup \{6\}} = \{1,2,4,6\}$

(E) goto (G)

(G) $i \leftarrow 5$

(B) —

(C) —

(D) $A_3 \leftarrow \overline{\{2,4\} \cup \{5\}} = \{1,2,4,5\}$

(E) goto (G)

(G) $i \leftarrow 4$

(B) goto (G)

(G) $i \leftarrow 3$

(B) —

(C) $r \leftarrow 1$; goto (C)

(C) —

(D) $i_2 \leftarrow 3$; $A_2 \leftarrow \overline{\{2\} \cup \{3\}} = \{2,3,4\}$

(E) —

(F) $r \leftarrow 2$; return

The actual values are now

$$r = 2$$

$$i_0 = 0,\ i_1 = 2,\ i_2 = 3$$

$$A_0 = \emptyset,\ A_1 = \{2\},\ A_2 = \{2,3,4\}\ .$$

The lectic successor of the subtree $\{2,4\}$ thus is the subtree $A_r = \{2,3,4\}$. Because of $A_2 = A_1 \oplus 3$

$$A_1 <_3 A_2\ ,$$

the output list also satisfies the conditions of 1.7 and can be used for another call of the algorithm.

In fact, understanding the structure of the I/O - list in 1.7 is the crucial step in proving the correctness of the algorithm. Let A be the closed set whose successor is to be computed, i.e. $A_r = A$ at the start. The conditions in 1.7 imply that, for each $j \leq r$

$$A_j = (\ldots((\overline{\emptyset} \oplus i_1) \oplus i_2)\oplus\ldots)\oplus i_j$$

and therefore $$A_j = \overline{A \cap \{1,2,\ldots,i_j\}} \ .$$

Steps (B) and (C) cause that $i_r < i$, therefore in Step (D)

$$\overline{A_r \cup \{i\}}$$

is equal to $A_r \oplus i$, and steps (E) and (F) can be jointly reformulated to

$$\underline{\text{If}} \quad A <_i A_{r+1} \quad \underline{\text{then}} \quad r \leftarrow r+1 \ ; \ \underline{\text{return}}.$$

Since $A_{r+1} = A \oplus i$, this is exactly the same as step (B) of 1.5. We also find that

$$A_r \oplus i = A \oplus i.$$

This guarantees that the ouput list is of the appropriate form.

One might wish to start the algorithm with some set different from $\overline{\emptyset}$. For this it is neccessary to build up an input list as required by 1.7.

1.9 PROPOSITION: For $A \subseteq M$ define numbers $r, i_0, i_1, \ldots, i_r$ and sets $A_0, A_1, \ldots, A_r$ by the following rules:

1) $i_0 := 0$, $A_0 := \overline{\emptyset}$

2) If i_j and A_j are already defined then either $A \setminus A_j = \emptyset$, in which case we define $r := j$

or else let

i_{j+1} := smallest element of $A \setminus A_j$

$A_{j+1} := \overline{A_j \cup \{i_{j+1}\}}$.

Then $A_r = \overline{A}$, and the so defined data list satisfies the conditions of 1.7.

Proof: Since $A_{j+1} = \overline{A_j \cup \{i_{j+1}\}}$ for $0 \leq j < r$, and since i_{j+1} is the smallest element of $A \setminus A_j$, we conclude that

$$A_o \subset A_1 \ldots \subset A_r$$
$$i_o < i_1 < \ldots < i_r .$$

Since $A_j = \overline{\{i_1, i_2, \ldots, i_j\}}$, $0 \leq j \leq r$
we conclude that

$$A_{j+1} = \overline{A_j \cap \{1,2,\ldots,i_j\} \cup \{i_{j+1}\}}$$

and $$A_j <_{i_{j+1}} A_{j+1} .$$

This shows that the conditions of 1.7 are satisfied. Clearly $\{i_1, i_2, \ldots, i_r\} \subseteq A$, thus $A_r = \overline{\{i_1, i_2, \ldots, i_r\}} \subseteq \overline{A}$. But $A \setminus A_r = \emptyset$, thus $A_r = \overline{A}$.

The practical value of 1.9 is that it provides an algorithm to obtain a suitable input list for 1.7. We need this when we want to compute all closed sets which are lectically larger than a given one.

A final remark: If the algorithm 1.7 is implemented using a programming language that admits "local" variables then the legendary I/O-list can be reduced to A_r, plus a logical flag to indicate the beginning and the end of the lectic list.

§2 PSEUDO-CLOSED SETS

Throughout this chapter, let M be a finite set and let

$$\overline{} : 2^M \to 2^M$$

be a closure operator. Our aim is to represent this operator by means of implications. To make this precise, define an IMPLICATION

(WITHIN M) to be a pair A,B of subsets of M, denoted $A \to B$. An implication $A \to B$ HOLDS for the given closure operator $^-$ if $B \subseteq \overline{A}$. A set J of implication holds if each of these implications holds.

Conversely, let J be a set of implications within M. A set $X \subseteq M$ RESPECTS J if for every implicaton $A \to B$ in J $A \subseteq X$ implies $B \subseteq X$. A closure operator respects J if every closed set does. There is a unique finest closure operator respecting a given set of implications, as the following proposition shows:

2.1 PROPOSITION: Let J be a set of implications within M. Then

$$\{X \subseteq M \mid X \text{ respects } J\}$$

is a closure system.

Proof: M obviously respects every set of implications. So all we have to show is that

$$\{X \subseteq M \mid X \text{ respects } J\}$$

is closed under intersections. Suppose $\mathcal{S}$ is a subfamily, i.e. $\mathcal{S} \subseteq \{X \subseteq M \mid X \text{ respects } J\}$, and let $S := \bigcap \mathcal{S}$. If $A \to B$ is some implication from J with $A \subseteq S$ then $A \subseteq X$ for every $X \in \mathcal{S}$, thus $B \subseteq X$ for every $X \in \mathcal{S}$, thus $B \subseteq S$.

$A \to B$ is a CONSEQUENCE of a set J of implications if every set which respects J also respects $A \to B$. An equivalent condition is that $A \to B$ holds for every closure operator for which J holds. A set of implications is REDUCED if none of these implications is a consequence of the others.

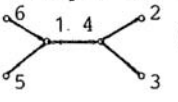
A set J of implications is COMPLETE (for a closure operator $\overline{}$) if the implications that hold for $\overline{}$ are exactly the consequences of J.

A QUASI-CLOSED SET is a set $Q \subseteq M$ such that $X \subseteq Q$ implies $\overline{X} = \overline{Q}$ or $\overline{X} \subseteq Q$ for all X. Clearly every closed set is quasi-closed.
A PSEUDO-CLOSED SET is a quasi-closed set which is not closed and satisfies a certain minimality condition. More precisely, $P \subseteq M$ is pseudo-closed if and only if

i) P is quasi-closed,

ii) P is not closed,

iii) $Q \subseteq P$ and $\overline{Q} = \overline{P}$ for quasi-closed Q implies $Q = P$.

2.2. EXAMPLE: For the closure operator of example 1.6 (where the closed sets are the subtrees of the tree 

) there are exactly 10 pseudo-closed sets. These are the two-element subsets which are not closed, like $\{1,2\}$, $\{1,3\}$, $\{2,3\}$ etc.

DUQUENNE & GUIGUES have given, in a different setting, a theorem that analyzes the rôle of the pseudo-closed sets:

2.3 THEOREM (cf. DUQUENNE & GUIGUES [2]):
The set of implications

$$\{P \to \overline{P} \mid P \text{ pseudo-closed}\}$$

is reduced and complete.

Proof: Call this family of implications J. We have to show that J is reduced and complete. It is trivially true that J holds for $\overline{}$, thus to show that J is complete we must prove that

every set which respects J is closed under $^{--}$. Let Q be a minimal counterexample, i.e. suppose that Q respects J, that $Q \neq \overline{Q}$, and that every proper subset of Q which respects J is closed. We claim that Q is quasi-closed. To see this consider some $X \subseteq Q$. By 2.1 there is a smallest set $\tilde{X} \supseteq X$ which respects J. Since Q respects J we have $\tilde{X} \subseteq Q$, and since $\overline{X}$ respects J we have $\tilde{X} \subseteq \overline{X}$. Thus if $\tilde{X} = Q$ then $\overline{X} = \overline{Q}$. On the other hand if $\tilde{X} \neq Q$ then, by the minimality assumption, $\tilde{X} = \overline{X}$ and thus $\tilde{X} \subseteq Q$. This shows that Q is quasi-closed.

But this is impossible: Q is not closed, so there has to be some pseudo-closed set P with $P \subseteq Q$ and $\overline{P} = \overline{Q}$, and $P \to \overline{P}$ is in J. Obviously, Q does not respect this implication.

To see that J is reduced, let P be some pseudo-closed set. We prove that P respects $J \setminus \{P \to \overline{P}\}$: If $A \subsetneq P$ is any pseudo-closed set then $\overline{A} \neq \overline{P}$. But since P is quasi-closed, this implies $\overline{A} \subseteq P$.

Is there a good algorithm to construct the pseudo-closed sets? Actually, we do not know. All we can offer are procedures that construct the closed and the pseudo-closed sets, or even all quasi-closed sets. These algorithms rest on the following proposition:

2.4 PROPOSITION:

1) The family of all quasi-closed sets is a closure system.
2) The family consisting of all sets which are either closed or pseudo-closed is a closure system.

The proof of 2.4 will be given after proposition 2.5. Note that 2.4 enables us to use the algorithms of §1 to generate all quasi-closed sets or, resp., all sets which are closed or pseudo-closed.

The missing tool is a method for computing the "quasi-closure" or the "pseudo-closure" of arbitrary sets $A \subseteq M$. This will also be implicitly given by 2.5.

2.5. PROPOSITION:

1) A set $Q \subseteq M$ is quasi-closed if and only if $Q \cap C$ is closed for every closed set C with $Q \not\subseteq C$.
2) A set $R \subseteq M$ is closed or pseudo-closed if and only if $\overline{P} \subseteq R$ for every pseudo-closed set $P \overset{C}{\neq} R$.

Proof: 1), "=>". Let Q be quasi-closed and let $Q \cap C =: D$, where C is some closed set with $Q \not\subseteq C$. Then $\overline{D} \subseteq \overline{C}$, and this implies $\overline{D} \neq \overline{Q}$, therefore $\overline{D} \subseteq Q$. Thus $\overline{D} = D$.
1), "<=". Suppose $Q \cap C$ is always closed when C is closed and $Q \not\subseteq C$. Let $A \subsetneq Q$ with $\overline{A} \neq \overline{Q}$. Then $\overline{A} \cap Q$ is closed, hence $\overline{A} \cap Q = \overline{A}$, i.e. Q is quasi-closed.
2), "=>". If R is pseudo-closed then R is quasi-closed, but not closed. This implies that $\overline{P} \subsetneq R$ for every set $P \subsetneq R$ with $\overline{P} \neq \overline{R}$. But if $P \overset{C}{\neq} R$ is pseudo-closed then clearly $\overline{P} \neq \overline{R}$.
2), "<=". Suppose R is not pseudo-closed and satisfies the condition, i.e. $\overline{P} \subsetneq R$ for every pseudo-closed set $P \subsetneq R$. Then R respects

$$\{P \to \overline{P} \mid P \text{ pseudo-closed}\}$$

and by 2.1, 2.2 therefore is closed.

The proof of 2.4 now reduces to showing that the characteristic properties described in 2.5 are preserved by arbitrary intersections. This is straightforeward in both cases, we give the second case as an example: Let X be some index set and let all elements of

$\{R_x \mid x \in X\}$ be closed or pseudo-closed. We have to show that the same holds for $R := \bigcap_{x \in X} R_x$. Let $P \subsetneq R$ be some pseudo-closed set. Then $P \subsetneq R_x$ for all $x \in X$, thus $\overline{P} \subseteq R_x$ for all $x \in X$, thus $\overline{P} \subseteq R$.

Propositions 2.4 and 2.5 can in several different way be utilized to construct algorithms for computing the pseudo-closed sets. Since we do not have much experience with these algorithms, and since we still hope for a better idea to construct these sets, we only give an example:

Using 2.5.2 we can decide if a set $X \subseteq M$ with $X = \overline{X}$ is pseudo-closed if we know all pseudo-closed sets properly contained in X, and their closures. Since $P \subseteq X$ implies that P is lectically smaller than X, the knowledge of all pseudo-closed sets X suffices to decide if X is pseudo-closed, and also to construct the lectically smallest pseudo-closed set $\nearrow X$. The following algorithm is a recursive application of this construction.

2.6 ALGORITHM to compute all sets which are pseudo-closed with respect to a given closure operator $\overline{}$ on the finite set $M := \{1,2,\ldots,m\}$.

Variables (for the dimension of the arrays see the remark below):

n is the number of pseudo-closed sets.

P(1), P(2),...,P(n) is the list of pseudo-closed sets.

$\overline{P}(1)$, $\overline{P}(2)$,...,$\overline{P}(n)$ is the list of the closures of the P(i).

LOG(1),LOG(2),...,LOG(n) is an auxiliary logical vector to avoid unneccessary work.

Algorithm:

```
(A)  n ← 0; A ← ∅; LOG(1) ← TRUE
(B)  If  ∅̄ ≠ ∅  then  n ← 1; P(1) ← ∅; P̄(1) ← ∅̄
(C)  For  i = m,m-1,...,1 step-1
(D)  If  i∈A  then goto (S)
(E)  B ← A⊕{i}; C ← B̄; (LOG(1),...,LOG(n)) ← (FALSE,...,FALSE)
(F)  FLAG ← FALSE
(G)  For  j = 1,2,...,n
(H)  If LOG(j) = TRUE then goto (O)
(I)  If  P(j)⫋C  then goto (N)
(J)  If  P(j)⫋B  then goto (O)
(K)  If  P̄(j)⊆B  then goto (N)
(L)  If  P̄(j)\A  contains an element  < i  then goto (S)
(M)  FLAG ← TRUE;  B ← B∪P̄(j)
(N)  LOG(j) ← TRUE
(O)  Next j
(P)  If FLAG = TRUE then goto (F)
(Q)  If  B = C  then  A ← C; goto (C)
(R)  n ← n+1;  P(n) ← B; P̄(n) ← C; A ← B; goto (C)
(S)  Next i
(T)  End
```

Remarks:

How much storage space should be reserved for the arrays $P,\overline{P},LOG$? In other words, how many pseudo-closed sets do we expect? We do not know the exact maximum, but it is easy to give examples with as many as $\binom{m}{\frac{m}{2}}$ pseudo-closed sets. However, the full list of implications seems to be of little interest in these examples. In practice one will therefore give a "reasonable" dimension to the arrays $P,\overline{P}$ and LOG .

If there are too many pseudo-closed sets there is also the possibility to compute only those pseudo-closed sets whose cardinality does not exceed some fixed number $C_{max.}$ This can be achieved by introducing an extra line after line (M) of the algorithm:

(M1) If $|B| > C_{max}$ then goto (S) .

Note that lines (C),(D),(L),(S),(T) of 2.6
correspond to (A),(B1),(B3),(C),(D) of 1.5.
Lines (E) — (P) correspond to (B2) of 1.5,
there the pseudo-closure is computed. The computation is based on prop. 2.4.2.

§ 3 APPLICATIONS IN CONCEPT ANALYSIS

Formal concept analysis has been introduced by WILLE as a mathematical tool for conceptual data analysis. We recall the basic notions in short:

A CONTEXT is a triple (G,M,I), where G and M are sets (of OBJECTS and ATTRIBUTES, resp.) and where I is a relation $I \subseteq G \times M$. We read $(g,m) \in I$ as "the object g has the attribute m".

For $A \subseteq G$, we define

$$A' := \{m \in M \mid (g,m) \in I \text{ for all } g \in A\}$$

and dually for $B \subseteq M$

$$B' := \{g \in G \mid (g,m) \in I \text{ for all } m \in B\}.$$

A pair (A,B) is a CONCEPT of (G,M,I) if $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. If (A,B) is a concept then A is called the EXTENT and B then INTENT of (A,B).

Clearly any concept (A,B) of (G,M,I) is uniquely determined both by its extent A and by its intent B. Moreover, a set $A \subseteq G$ is an extent of some concept of (G,M,I) if and only if $A = B'$ for some $B \subseteq M$, and this is equivalent to $A = A''$. It is therefore not ambiguous to call such a set A an extent of (G,M,I). Dually, an intent of (G,M,I) is a set $B \subseteq M$ satisfying $B = B''$.

3.1. EXAMPLE: A small context (G,M,I) can conveniently be represented by a rectangular $G \times M$-table, with crosses indicating the relation I. In fig. 2 $G = \{1,2,3,4\}$, $M = \{a,b,c,d,e\}$ and, e.g. $(1,a) \in I$ but $(1,c) \notin I$. In this example $(\{1,4\}, \{b,d\})$ is a concept while $(\{1,2\}, \{b\})$ is not.

	a	b	c	d	e
1	X	X		X	
2		X	X		
3	X			X	X
4		X		X	

fig. 2

The starting point of formal concept analysis is the observation that the two mappings

$$': 2^G \to 2^M \quad \text{and} \quad ': 2^M \to 2^G$$

form a Galois-connection. As an immediate consequence one obtains that the family of all extents and the family of all intents of (G,M,I) both are closure systems, the corresponding closure operators are the mappings $X \to X''$ on G and M, resp.

One of the basic tasks of concept analysis is the generation of all concepts of a given context. This problem has been looked at,

in different terms, by several authors (e.g. FAY[1], NORRIS[3]).

Since a concept is uniquely determined by its intent and since the intents form a closure system, we can apply the results of § 1 to determine all concepts. The following algorithm is a version of 1.7, modified for our special purpose:

2.2 ALGORITHM "NEXT CONCEPT"

Purpose: Computes the successor of any given concept (A,B) of the context (G,M,I). Here $M = \{m_1, m_2, \ldots, m_m\}$, and the concepts are ordered lectically by their intents.

Input/Output:

A number $r \geq 0$

a list $i_0 < i_1 < \ldots < i_r$ of numbers (all $\leq m$)

a list $A_0 \supset A_1 \supset \ldots \supset A_r$ of extents

a list $B_0 \subset B_1 \subset \ldots \subset B_r$ of intents

satisfying the following conditions:

$$i_0 = 0,\ A_0 = G,\ A_r = A$$

$$A_j' = B_j,\ B_j' = A_j \quad \text{for} \quad j = 0,1,2,\ldots,r$$

$$B_j = (B_{j-1} \cup \{m_{i_j}\})'' \quad \text{for} \quad j = 1,2,\ldots,r$$

m_{i_j} is the smallest element of $B_j \setminus B_{j-1}$ } for $j = 1,2,\ldots,r$

Algorithm:

(A) For $i = m, m-1, \ldots, 1$ step -1

(B) If $m_i \in B_r$ then goto (P)

(C) If $i < i_r$ then $r \leftarrow r-1$; goto (C)

(D) $i_{r+1} \leftarrow i$; $A_{r+1} \leftarrow A_r \cap \{m_i\}'$

(E) For $j = 1, 2, \ldots, i-1$

(F) If $\{m_j\}' \supseteq A_{r+1}$, then goto (P)

(G) Next j

(H) $B_{r+1} \leftarrow B_r \cup \{m_i\}$

(I) For $j = i+1, i+2, \ldots, m$

(J) If $j \in B_{r+1}$ then goto (L)

(K) If $\{m_j\}' \supseteq A_{r+1}$ then $B_{r+1} \leftarrow B_{r+1} \cup \{m_j\}$

(L) Next j

(M) $r \leftarrow r+1$

(N) Return

(P) Next i

(Q) Stop: No successor (input was $B_r = M$).

Remarks:

The input list and the output list have the same structure. If the input satisfies the above conditions then the output automatically does and can be used for the next input (except when stop was reached). The pair (A_r, B_r) of the output list is the next concept. If the program is initially called with

$$r = 0 = i_o,\ A_o = G,\ B_o = G'$$

then successive runs lead through all concepts of (G,M,I).

To obtain a correct input list for computing the successor of an arbitrary concept (A,B), compute (cf. 1.9):

$$i_o := 0,\ A_o := G,\ B_o := G'$$

and, recursively

$$\text{if } B \setminus B_j = \emptyset \text{ then } r := j$$

$$\text{else } i_{j+1} := \min\{x \mid m_x \in B \setminus B_j\}$$

$$A_{j+1} := (B_j \cup \{m_{i_j}\})'$$

$$B_{j+1} := A'_{j+1} \cdot$$

3.3 EXAMPLE: The concepts of the context in example 3.1 as produced by 3.2:

The first input and also the first concept is

$$A_o = \{1,2,3,4\},\ B_o = \emptyset,\ r = 0$$

The later values are:

Concept no.	r	i_1 i_2 i_3	A_1 A_2 A_3	B_1 B_2 B_3
2	1	4	{1,3,4}	{d}
3	1	3	{2,3}	{c}
4	1	2	{1,2,4}	{b}
5	2	2,4	{1,2,4},{1,4}	{b},{b,d}
6	2	2,3	{1,2,4},{2}	{b},{b,c}
7	1	1	{1,3}	{a,d}
8	2	1,3	{1,3},{3}	{a,d},{a,c,d,e}
9	2	1,2	{1,3},{1}	{a,d},{a,b,d}
10	3	1,2,3	{1,3},{1},∅	{a,d},{a,b,d},{a,b,c,d,e}

Martin Skorsky has used this algorithm to compute concept systems with approximately 10^6 concepts.

Another fundamental problem is to study the "logic" of a context, i.e. to describe the dependencies between the attributes. A typical such dependency would be "every object having the attributes a,b,c... also has the attributes x,y.z,...". This is equivalent to $\{x,y,z,...\} \subseteq \{a,b,c...\}$". In other words it is the same as the implication $\{a,b,c...\} \to \{x,y,z,...\}$ for the closure system of the intents.
We have prepared for this in §2, where the pseudo-closed sets were introduced to describe the implications. Spezialized to our purpose, the definitions are:

A QUASI-INTENT of a context (G,M,I) is a set $Q \subseteq M$ such that $X \subseteq Q$, $X'' \neq Q''$ implies $X'' \subseteq Q$, for all $X \subseteq M$.

A PSEUDO-INTENT is a set $P \subseteq M$ such that

1) P is a quasi-intent

ii) P is not an intent

iii) If $Q \subseteq P$ is a quasi-intent with $Q'' = P''$ then $Q = P$.

It was shown in 2.3 that the set of implications

$$\{P \to P'' \mid P \text{ pseudo-intent}\}$$

is a complete and reduced set, i.e. a minimal set from which all other implications follow. From proposition 2.5 we conclude a method for deciding whether a given set is a quasi-intent or a pseudo-intent:

3.4 PROPOSITION:

1) A set $Q \subseteq M$ is a quasi-intent of (G,M,I) if and only if $Q \cap \{g\}' = (Q \cap \{g\}')''$ for every $g \in G$ with $\{g\}' \not\subseteq Q$.
2) A set $P \subseteq M$ is a pseudo-intent of (G,M,I) if and only if it is not an intent and satisfies for all $R \subseteq M$ the following: If R is a pseudo-intent and properly contained in P, then $R'' \subseteq P$.

The set of all pseudo-intents can be computed with algorithm 2.6.

A particularly interesting application occurs if the context is not explicitely available because the number of objects is too large. We then use a version of algorithm 2.6 which allows the user to enlarge the context while the algorithm is running. The result is a program for a kind of "computer-aided theorem proving".

We shall not give this program in detail. The reader who has mastered the details of algorithm 2.6 will have no difficulties to write the algorithm after reading the following example, due to R.WILLE.

The objects in this example are the binary relations, i.e. subsets $R \subseteq S \times S$, where S is an arbitrary set. We consider the following properties that binary relations may or may not have:

	name	definition
r	reflexive	xRx for all $x \in S$
i	irreflexive	$\neg\, xRx$ for all $x \in S$
s	symmetric	$xRy \Rightarrow yRx$ for all $x,y \in S$
as	asymmetric	$xRy \Rightarrow \neg\, yRx$ for all $x,y \in S$
an	antisymmetric	xRy and $yRx \Rightarrow x=y$ for all $x,y \in S$
t	transitive	xRy and $yRz \Rightarrow xRz$ for all $x,y,z \in S$
nt	negatively tr.	$\neg\, xRy$ and $\neg\, yRz \Rightarrow \neg\, xRz$ for all $x,y,z \in S$
c	complete	xRy or yRx for all $x \neq y \in S$
sc	strongly compl.	xRy or yRx for all $x,y \in S$

What are the implications that hold between these properties? For a single implication it is not difficult to decide if it holds or not: We can either prove it or give a counterexample. There are only finitely many possible implications, but far too many to test them all. What we aim for is an efficient way of finding counterexamples and "good" implications.

As long as we have no examples, all implications are possible. Therefore we start by considering all relations on a one-element and on a two-element set. There are, up to isomorphism, twelve such relations:

S	R	r	i	s	as	an	t	nt	c	sc	
{0}	$\varnothing$		X	X	X	X	X	X	X		
{0}	{(0,0)}	X		X		X	X	X	X	X	
{0,1}	$\varnothing$		X	X	X	X	X	X			
{0,1}	{(0,0)}			X		X	X				←
{0,1}	{(0,0),(1,1)}	X		X		X	X				
{0,1}	{(0,0),(0,1)}					X	X	X	X		←
{0,1}	{(0,0),(1,0)}					X	X	X	X		←
{0,1}	SxS \ {(0,1)}	X				X	X	X	X	X	
{0,1}	SxS \ {(0,0)}			X				X	X		←
{0,1}	{(0,1)}		X		X	X	X	X	X		
{0,1}	{(0,1),(1,0)}		X	X				X	X		
{0,1}	SxS	X		X			X	X	X	X	

Now we have a context to start with (in general, this context may be empty). Of course, only those implications can hold for all binary relations that hold for this context, but the converse is not true. Note that four of our examples are superfluous (those indicated with the arrows) because their intents are obtainable as intersections of other intents and therefore respect all implications which the others respect. We shall omit these in the sequel.

Now we use algorithm 2.6 to compute the pseudo-intents. The lectically first pseudo-intent turns out to be $\{sc\}$, with $\{sc\}'' = \{r,t,nt,c,sc\}$. In other words, the implication

$$\{sc\} \to \{r,t,nt,c,sc\}$$

holds in all our examples. Does it hold in general? Of course not. A counterexample is, e.g. $S = \{0,1,2\}$, $R = SxS \setminus \{(0,1),(1,2),(2,0)\}$.

This relation is reflexive, antisymmetric, complete and strongly complete and has none of the other properties. We include this example in our context and ask again for the smallest pseudo-intent. Stll, this is $\{sc\}$, but now $\{sc\}'' = \{r,c,sc\}$, and we have to check if the implication $\{sc\} \to \{r,c,sc\}$, which we abbreviate to

$$sc \to r,c,$$

holds for all relations. It clearly does, every strongly complete relation is complete and reflexive.

The next pseudo-intent produced by 2.6 is $\{t,c\}$, with $\{t,c\}'' = \{t,nt,c\}$ This suggests the implication

$$t,c \to nt,$$

which indeed holds for binary relations. The same is true for the next suggested implication

$$an,nt \to t,$$

which comes from the pseudo-intent $\{an,nt\}$ with $\{an,nt\}'' = \{an,t,nt\}$. Then we obtain the pseudo-intent $\{as\}$, and $\{as\}'' = \{i,as,an,t,nt\}$ in our context of examples. But the implication

$$as \to i,an,t,nt$$

does not hold in general, as the following example shows:

Let $S = \{0,1,2\}$, $R = \{(0,1),(1,2),(2,0)\}$. This relation has the properties i,as,an,nt, and we include it in our list.

Since we have just changed our context of examples, one might think that we have to start algorithm 2.6 from the beginning. Fortunately this is not necessary, according to the following proposition:

3.4 PROPOSITION:

Let g be an object of the context (G,M,I) and denote by (G_g,M,I_g) the context obtained from (G,M,I) be deleting the object g, i.e. $G_g := G \setminus \{g\}$, $I_g := I \cap (G_g \times M)$.
Furthermore, let $A \subseteq M$ be some set.
If $\{g\}'$ respects all implications of the form $P \to P''$, where P is a pseudo-intent of (G_g,M,I_g) which is lectically smaller than A, then the pseudo-intents of (G,M,I) which are smaller than A are the same as those of (G_g,M,I_g).

Proof: Consider the smallest counterexample and apply 2.5.2 to obtain a contradiction.

By this proposition no new pseudo-intent less than our current one can occur. Therefore we can continue with algorithm 2.6 for the changed context. Still $\{as\}$ is an undecided pseudo-intent, but now, in the extended context, $\{as\}'' = \{i,an,as\}$, which suggests the implication

$$as \to i,an$$

which in fact holds for all relations.
The next two implications

$$s,c \to nt$$

$$s,an \to t$$

also hold, then

$$i,t \to as,an,nt$$

can be disproved by a counterexample, etc.
The complete result is given in the following tables.

A complete list of examples:

No	S	R
1	{0}	∅
2	{0}	{(0,0)}
3	{0,1}	∅
4	{0,1}	{(0,0),(1,1)}
5	{0,1}	S×S ∖ {(0,1)}
6	{0,1}	{(0,1)}
7	{0,1}	{(0,1),(1,0)}
8	{0,1}	S×S
9	{0,1,2}	S×S ∖ {(0,1),(1,2),(2,0)}
10	{0,1,2}	{(0,1),(1,2),(2,0)}
11	{0,1,2}	{(0,1)}
12	{0,1,2}	{(0,1),(1,0)}
13	{0,1,2}	S×S ∖ {(0,1)}
14	{0,1,2}	S×S ∖ {(0,1),(1,0)}

The reduced context of examples

	r	i	s	as	an	t	nt	c	sc
1		x	x	x	x	x	x	x	
2	x		x		x	x	x	x	
3		x	x	x	x	x	x		
4	x		x		x	x			
5	x				x	x	x	x	x
6		x		x	x	x	x	x	
7		x	x				x	x	
8	x		x			x	x	x	x
9	x				x			x	x
10		x		x	x			x	
11		x		x	x	x			
12		x	x						
13	x						x	x	x
14	x		x						

A reduced and complete list of implications

sc → r,c
t,c → nt
an,nt → t
as → i,an
s,c → nt
s,an → t
i,t → as,an
i,an → as
i,s,as,an,t → nt
r,c → sc
r,nt → c,sc
r,s,nt,c,sc → t
r,i → all properties.

Note that the premises of the above implications are exactly the pseudo-intents of the context.

It is not guaranteed by our method that the list of counterexamples is reduced (as it is in this example), the addition of new examples may make previous examples superfluous. The elimination of reducible examples has no influence on the implications.

REFERENCES:

[1] G.FAY, An algorithm for finite Galois connections. Technical report, Institute for Industrial Economy, Budapest(1973).

[2] J.-L. GUIGUES and V. DUQUENNE, Informative implications derived from a table of binary data. Preprint, Groupe Mathématiques et Psychologie, Université René Descartes, Paris (1984).

[3] E.M.NORRIS, An algorithm for computing the maximal rectangles in a binary relation. Rev. Roum. Math. Pures et Appl., Tome XXIII, No 2, pp 243-250, Bucarest (1978).

[4] R.WILLE, Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. RIVAL (ed.), Ordered sets. Reidel Publ. Comp., Dordrecht-Boston (1982), pp 445-470.

Technische Hochschule Darmstadt
Fachbereich Mathematik
Forschungsgruppe Begriffsanalyse
D-6100 Darmstadt , West Germany.

Author Index